VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Webová aplikace pro inteligentní práci s položkami vzdálených webových serverů

Web Application for Intelligent Management of Entities of Remote Web Servers

2011 Petr Pospěch

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra měřicí a řídicí techniky

Zadání bakalářské práce

Student: **Petr Pospěch**

Studijní program: B2649 Elektrotechnika

Studijní obor: 2601R004 Měřicí a řídicí technika

Téma: Webová aplikace pro inteligentní práci s položkami vzdálených webových serverů
Web Application for Intelligent Management of Entities of Remote Web Servers

Zásady pro vypracování:

Cílem práce je vyvinout a otestovat řešení webové aplikace pro monitorování vzdálených webových serverů a intelintní práci s jejich položkami.

Práce bude obsahovat tyto body:
1. Zpracování teoretické problematiky práce s webovými servery prezentujícími položky/data formou webových stránek. Problematika HTML parserů a XML.
2. Návrh a realizace webové aplikace pro vyhledávání nových/změněných položek na vzdálených webových serverch s možností jejich dalšího zpracování. Aplikace bude disponovat rozhraním pro definici uživatelského nastavení a zároveň i možností zasílání vybraných aktualizovaných položek ze vzdálených monitorovaných serverů zvoleným komunikačním kanálem (minimálně e-mail).
3. Otestování vyvinutého řešení na několika vybraných vzdálených serverech. Testování bude probíhat na vzdáleném serveru, na kterém se bude každou hodinu hledat nějaká položka a v případě výskytu nové/aktualizované se pošle email uživateli s odkazem na tuto položku včetně krátkého popisu položky, případně i jejího fota či dalších doplňkových informací relevantních k danému zaměření serveru.
4. Diskuze dosažených výsledků.
5. Extended abstrakt v anglickém jazyce v šabloně dle vedoucího práce v rozsahu 6 stran (šablona je k dispozici u vedoucího práce). Extended abstrakt bude do vlastní práce vložen jako příloha č.1.

Seznam doporučené odborné literatury:

1. KAČMÁŘ, D. *Programujeme .NET aplikace ve Visual Studiu .NET.* 1. vyd. Praha: Computer Press, 2001. 335 s. ISBN 80-7226-569-5.
2. HERNANDEZ,M. J. - VIECAS,J. L. *Myslíme v jazyku SQL : tvorba dotazů : knihovna programátora.* 1. vyd. Praha: Grada Publishing, 2004. 378 s. ISBN 80-247-0899-X.
3. WIGLEY, A. - SUTTON, M. - WHEELWRIGHT, S. *Microsoft .NET Compact Framework.* Redmond(USA): Microsoft Press, c2004. 860 s. ISBN 0-7356-1725-2 .
4. LACKO, l. *Programujeme mobilní aplikace ve Visual Studiu .NET.* 1. vyd. Brno: Computer Press, 2004. 479 s. ISBN 80-251-0176-2.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Ondřej Krejcar, Ph.D.**

Datum zadání: 19.11.2010

Datum odevzdání: 06.05.2011

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení Studenta

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

Datum Podpis

Poděkování

Děkuji vedoucímu práce Ing. Ondřeji Krejcarovi, Ph.D. za pomoc při tvorbě bakalářské práce.

Abstrakt

Na webových serverech (obchody, aukce, datová úložiště) se neustále objevují nové položky, a v některých případech je třeba hlídat tyto položky. Tento projekt se tedy zabývá právě řešením tohoto problému formou uživatelské aplikace pro sledování a monitorování položek na webových serverech. Při každém dotazu na vyhledávání odpoví webový server s několika výrazy, které odpovídají zadanému hledanému výrazu. Všechny položky s odpovídajícími výrazy se uloží a pošlou uživateli zvoleným komunikačním kanálem, nejčatěji emailem. Tato problematika je řešena ve vytvořené aplikaci pro windows, která prohledává položky na serverech a informuje o nových položkách např. emailem. Aplikace obsahuje definici uživatelského prostředí a možnost zasílání informací emailem.

Klíčová slova

web server; C# aplikace; .NET framework; vyhledávání položek; webová aplikace

Abstract

On some web sites (shops, auctions, data storage) are constantly created new items, and in some cases, it is necessary to watch these items. This project deals a solution to this problem via a custom application to track and monitor items on the web servers. For each query to search, the web server responds with a few words that match the specified search term. All items matching expressions are saved and sent to the user selected communication channel common by email. This problem is solved in the created application for Windows that scans on the server and information on new items such as email. The application contains a definition for the user interface and the possibility of sending information by email.

Key Words

web server; C# application; .NET framework; items search; web application

Seznam použitých symbolů a zkratek

um – mikrometry, jednotka délky

PC – počítač

DLL - Dynamic-link library, dynamicky linkovaná knihovna

ms – milisekundy, jednotka času

C# - C sharp, programovací jazyk

Obsah

1 Úvod 1

2 Problematika vyhledávání na webových sídlech 2

2.1 HTML 2

2.2 Parsování v prohlížečích 3

2.3 Syntaktická analýza 3

2.4 Extensible Markup Language 4

2.5 Microsoft Visual Studio 2010 5

2.6 Programovací jazyk C# 6

3 Návrh aplikace pro monitoring vzdálených webových sídel 8

3.1 Struktura aplikace 11

3.2 Hlavní okno aplikace 12

3.3 HTML načtení kódu 15

3.4 HTML parser 16

3.5 Funkce check 21

3.6 Funkce pretty 22

3.7 Zasílání výsledků emailem 23

3.8 Funkce loadInfo 24

3.9 Časovač 24

3.10 Okno Settings 24

4 Případové studie 26

4.1 Monitoring ostravského ovzduší 26

4.2 Server priroda 26

4.3 Monitoring ovzduší ČR 27

5 Testování řešení 29

6 Diskuze dosažených výsledků 32

7 Závěr 34

1 Úvod

Na vzdálených webových serverech se často objevují nové položky, o kterých by chtěl být uživatel obeznámen. Vzhledem k velkému množství položek a vytížení by chtěl být uživatel obeznámen s vybraným typem a to při každém spuštění aplikace, případně neustále [1].

Uživatel má tedy aplikaci, která sama prohledává předem definovaný server (servery) a hledá na něm definované položky. Je vytvořena aplikace, při jejímž běhu jsou komunikačním kanálem(email) odesílány nové informace odpovídající hledání. Při stálém běhu aplikace je web server (servery) v určitém časovém intervalu dotazován (aby nedošlo k přetížení serveru), zda nebyla položka změněna. Při nalezení shody s definovanými termíny je uživatel informován emailem.Vybrané informace budou odeslány na definovaný email (emaily) a připraveny na zpracování uživatelem. [2].

Aplikace je vyvinuta v prostředí .Net Framework, ve vývojovém prostředí Microsoft Visual Studio 2010 v jazyku C#. Pro přístup aplikace na web servery je využito HTML tagů webové stránky, pomocí nichž se zjišťuje, zda na stránce došlo ke shodě s hledaným výrazem. Aplikace projde celou stránku serveru a při dohledání všech položek jsou získaná data zpracována a výsledek odeslán. Při stále spuštěné aplikaci jsou servery dotazovány v definovaném čase. V případě nalezení shody položek je informace ihned zpracována a odeslána. Důležité je dbát na to, aby servery nebyly přespříliš často dotazovány[3]. Dotazovat se po jedné sekndě je zcela zbytečné, neboť je využíván strojový čas dotazovaného serveru a velké množství informací by vedlo k zahlcení serveru a neposkytlo by v takto krátkých intervalech žádné nové informace. Prohledávání serverů stačí přibližně 1x za hodinu [4] jež je nastaveno jako přednastavená hodnota.

V druhé kapitole této práce nalezneme obecné informace o technologiích, které jsou v této práci použity. Bude předveden teoretický úvod do problematiky vyhledávání na webových sídlech, dozvíme se o HTML struktuře, syntaktické analýze a XML. Poté se budeme věnovat praktické stránce programu, dozvíme se o Visual studiu, v němž je aplikace vytvořena a jazyku C# v němž je napsána.

V třetí kapitole se budeme věnovat samotné aplikaci. Od návrhu aplikace z pohledu server-počítač přejdeme k samotnému návrhu programu. Po předvedení základního návrhu aplikace se budeme zabývat realizovanou aplikací. Bude ukázán rozdíl mezi návrhem a finální verzí. Následně budou probrány hlavní části aplikace. Hlavní důraz bude kladen na parser a funkce s ním spojené. Bude rozebráno použití DLL knihoven. V této kapitole budou popsány vlákna a email.

Čtvrtá kapitola se bude věnovat jednotlivým případovým studiím. Zde budou popsány všechny servery, které aplikace umí zpracovat. Bude zde srovnání mezi jednotlivými servery.

Pátá kapitola bude obsahovat testování aplikace.Budou zde popsány jednotlivé dryhy testojání jak a co se testuje.

2 Problematika vyhledávání na webových sídlech

Tato kapitola se bude věnovat problematice spojené s vyhledáváním a webovými sídly. Budou zde vysvětleny pojmy související s danou problematikou.

2.1 HTML

HTML, je zkratka pro **HyperText** Markup Language. **HyperText** Markup Language je v nynější době jazyk nejvíce používaný pro tvorbu a správku webových stránek. Značkovací jazyk je sada tzv. tagů. Tagy jsou vlastní značky, které jsou použity pro identifikaci zvláštních kombinací znaků v textu, jež uvozují akci překladače. HTML používá tagy pro identifikaci překladače pro pospání vzhledu webové stránky.

HTML je psáno tagy skládající se z uzavřených značek v hranatých závorkách (př. <h3>) a samotným obsahem webové stránky. V HTML se značkování používá často v párech, jako je <h3> a </ h3>. První ze značek uvozuje překladači, který spustí určitou fázi pro přeměnu textu, např. zvýraznění, změna velikosti nebo podtržení. Druhá značka podobná první, lišící se lomítkem, uvozuje překladač o ukončení změny formátování textu. Značky se mohou nazvat otevírací a uzavírací.
Př.: Povinné informace

Pro webový prohlížeč je úkolem číst HTML strukturu a složit ji do vizuální podoby textu. Webový prohlížeč HTML tagy nezobrazí, slouží pouze pro jeho formátování. Tento typ formátování je použit i naší aplikaci. Hypertext markup language umožňuje vložit do textového souboru obrázky a geometrické tvary pro vizualizační vzhled stránky a vytvořit podobně jako word tabulky a formuláře i s grafickým provedením. Umožní tedy vytváření dokumentů, obsahující sady nadpisů, různých fontů textu, změny velikosti a formátování textu, zarovnání textu, ale i grafických prvků. Umožní vkládání citací a rámečků a vložení scriptů jako PHP, javascript, které jako další kód umožní měnit vzhled a funkci samotného html kódu.
Př. Použití:

```
<skripttype="text/javascript" src="http://www.zuova.cz/ajax/mootools.js"></script>
<script type="text/javascript" src="http://www.zuova.cz/ajax/mootools-plugins.js"></script>
<script type="text/javascript" src="http://www.zuova.cz/ajax/slimbox.js"></script>
<script type="text/javascript" src="http://www.zuova.cz/ajax/dropdown.js"></script>
```

Značky bývají většinou párové, ale v jazyku HTML jsou i značky nepárové. Nepárové značky nemají žádný obsah, ani žádnou koncovou značku, která se používá pro ukončení. Příkladem může být komentář v kódu.

Pro snadnou práci prohlížečů s webovými stránkami umožní prohlížeč odkazovat se na kaskádové styly, tzv. CSS, která definuje vzhled a uspořádání textu v dokumentu. Jednotnou strukturu HTML kódu definuje W3C, které standarty CSS a HTML udržuje v jednotě.[5]
Jazyk HTML se stal od verze 2 součástí SGML. Pro tuto verzi jsou použity definované značky a jejich atributy. Např. značka se stala otevírací značkou pro zvýraznění textu a značkou koncovou. Součástí jazyka mohou být doplňující značky (elementy), které mají doplňující informativní charakter. Elementy popisují jejich vlastnosti, nenesou však jinou informaci.

Př. Elementu: <?xml version="1.0" encoding="windows-1250"?>

2.2 Parsování v prohlížečích

Webový prohlížeč je program, jehož účelem je čtení HTML kódu a složit ji do vizualizační podoby. Cílem je HTML kód načíst, správně rozpoznat uvozovací značky a elementy a převézt do vizualizační podoby pro přečtení uživatelem.

HTML kód při čtení rozpozná značky a provádí operaci syntaktická analýza, též známou jako parsování nebo parsing. Každý prohlížeč má svou tabulku značek, které rozpozná. Toto je důvodem, proč některé stránky vypadají na různých prohlížečích odlišně. Některé prohlížeče jako Opera, Firefox umožňují dokonce vytvoření vlastních značek a jejich použití.

Nejrozšířenější prohlížeč, Windows Internet Explorer, však toto nedovoluje a obsah neznámých elementů zobrazuje normálně, bez stylu, vlastní elementy se proto prakticky nepoužívají. Elementu je přiřazen styl (způsob zobrazení). Styly mohou být uvedeny ve stylovém předpisu. Vlastnosti neznámých stylů, které není schopen prohlížeč zpracovat, zobrazí podle defaultního stylu, který má zabudován. Některé prohlížeče umožňují uživateli implicitní styly definovat.[6]

2.3 Syntaktická analýza

Syntaktická analýza (slangově podle angličtiny též *parsování* nebo *parsing*) se v informačních technologiích a lingvistice navýzá proces, který pomocí formálních značek a elementů skládá HTML strukturu do vizualizační podoby podle předem daných pravidel.

Program, který toto vykonává se nazývá parser nebo syntaktický analyzér. Parsováním vstupního textu do přijatelné podoby, vhodné pro další zpracování, se získá text, vhodný pro další úpravu.

Prvním krokem pro parsování je analáza, při níž se ze vstupního textu vytvoří jistá posloupnost tzv. tokenů, které jsou nositelem informací na stránce. Token je tedy elementární nositel výrazu v rámci daného jazyka. Tokenem je např. závorka, znak, číslo, řetězec, sada znaků, symboly a pod. Pro parser je to již nejmenší datová jednotka, která je zpracována.
Existují programy, které z programovacího jazyka popsaného v Backus-Naurově notaci vytvořit příslušný parser, např. Yacc (*yet another compiler compiler*).

Parsování se využívá při kompilaci. Kompilátor nejdříve analyzuje HTML kód této webové stránky a vytvoří datovou strukturu vhodnou prodalší zpracovní, např. v naší aplikaci pro porovnávání. Programovací jazyk (HTML) je specifikován tagy, tedy značkami uvozující kód pro vizualizaci, parser tyty značky používá k naparsování dané stránky. Tyto parsery bývají programovány ručně, vzhledem k složitosti html kódu, ale také proto, že mohou být upraveny pro konkrétní výstup a usnadnit tak další programování a zpracování daného kódu.[7]

Aplikace, jako program na PC, běží pouze při spuštění, kdežto server běží kontinuálně, což znamená, že data, která přicházející na server nemají žádnou odezvu na náš systém. Cílem práce je najít způsob jakým by server komunikoval s PC a odesílal mu zvolená data v definovaném formátu.

Potřebujeme-li mít přístup na jiné servery, musela být nalezená jiná, univerzální metoda, která je platná pro všechny stránky. Vzhledem ke složitosti stránek a jejich různorodosti, aplikace musí zpracovat definovaný výraz v jakémkoli stylu, který bude potřeba. Jednou z přijatelně složitých metod je procházení kódu stránky. Řešením této aplikace je parsování.

2.4 Extensible Markup Language

Extensible Markup Language (zkráceně **XML**, česky rozšiřitelný značkovací jazyk) je obecný jazyk, jež byl vyvinut a standardizován konsorciem W3C. Jedná se o novou podobu staršího jazyka SGML. XML umožňuje vytvoření konkrétních značkovacích jazyků pro nejrůznější účely. Jazyk XML je podporován programovacích jazyků (HTML) a řadou nástrojů. Slouží pro výměnu dat mezi dokumenty, které popisují strukturu dokumentu z datového hlediska. Vizualizace dokumentu bývá zprostředkována pomocí CSS.[8]

Extensible Markup Language XML se stal defakto standardem pro výměnu informací a zastoupení na internetu. XML parsování je základní operace, prováděná na dokumentu XML, aby mohl být přístupný a manipulovatelný. Tato operace způsobují výkonnostní překážky v aplikacích a systémech, které zpracovávají velké objemy dat XML. Věříme, že podobnost je přirozený způsob pro zvýšení výkonnosti. Využití vícejádrových procesorů může nabídnout nákladově-efektivní řešení, protože budoucnost vícejádrových procesorů bude podporovat stovky jader, a nabídne vysoký stupeň paralelismu v hardwaru. [9]

Parsování XML stránky je možnost, jak zpracovat obsah webu bez ohledu, jestli používá nějakou službu jako RSS či nikoliv. Formát XML nám umožní vyčíst strukturu kódu stránky a libovolně ji zpracovávat. V naší aplikaci prohledáváme načtený kód zadané stránky a zjišťujeme typ položek, které se v něm vyskytují.

Příklad XML dokumentu:

```
<SHOP>
 <SHOPITEM>
   <PRODUCT>Kosmetické kapesníčky Mars - 100ks</PRODUCT>
   <DESCRIPTION>Extra jemné, třívrstvé,
     kosmetické kapesníčky značky
    Mars v praktické plastové krabičce.                    </DESCRIPTION>
   <URL>http://priklad.cz/kosmetika/kkmars100</URL>
   <IMGURL>http://priklad.cz/imgs/kkmars100.jpg</IMGURL>
   <PRICE>47</PRICE>
   <PRICE_VAT>56</PRICE_VAT>
 </SHOPITEM>
 <SHOPITEM>
   <MANUFACTURER>Mars</MANUFACTURER>
   <PRODUCT>Kosmetické kapesníčky Mars - 200ks</PRODUCT>
   <DESCRIPTION>Extra jemné, třívrstvé,
     kosmetické kapesníčky značky
    Mars v praktické plastové krabičce.                    </DESCRIPTION>
```

```
    <URL>http://priklad.cz/kosmetika/kkmars200</URL>
    <PRICE>73</PRICE>
    <PRICE_VAT>87</PRICE_VAT>
    <VAT>0.19</VAT>
   <CATEGORYTEXT>Kosmetika</CATEGORYTEXT>
    <IMGURL>http://priklad.cz/imgs/kkmars200.jpg</IMGURL>
  </SHOPITEM>
</SHOP>
```
[10]

Příklad XML dokumentu je velmi podobný tomu, jaký je zpracováván v naší aplikaci a jaký formát přibližně má. Výhodou tohoto formátu XML je fakt, že tento formát se přímo nabízí jako dokument k odeslání e-mailem. Jelikož aplikaci zajímá pouze samotný obsah, prohledává pouze vyšší instanci položek, popř. jeho první podstupeň, který dodává jakési dodatečné informace o položce, které jsou nutné k rozlišení položky. Po projetí celého kódu jsou položky odeslány e-mailem v pořadí, jakém byly načteny aplikací. V případě, že nedojde k úspěšnému odeslání je uživatel aplikací informován. Aplikace se znovu pokusí za stanovený interval spustit a načíst kód stránky tak, jak by tomu bylo při úspěšném pokusu. Uživatel má možnost vyřešit neúspěšný pokus manuálním spuštěním.

2.5 Microsoft Visual Studio 2010

Software Microsoft Visual Studio 2010 je nezbytným nástrojem pro základní úkoly vývoje aplikace a výrazně napomáhá programátorům realizovat jejich nápady. Software poskytuje pokročilé nástroje pro práci s textem, databázemi, čísly a obsáhlé ladící nástroje a automatickou nápovědu v podobě tipů při psaní programového kódu. Softwarem Microsoft Visual Studio 2010 lze díky výkonným nástrojům uvést myšlenky vývojářů v realizaci a dává velký prostor schopnostem a fantazii programátora. Software Visual Studio 2010 přináší velká vylepšení: návrháře pro rychlejší vývoj, podstatná zdokonalení vývojových nástrojů a zkodonalení jazyka, která zrychlují vývoj pro data všech typů. Microsoft Visual Studio 2010 přináší rychlejší psaní kódu díky nápovědě, která se snaží doplnit za vývojáře dopisování vlastního kódu díky propracovanému systému zpracování proměnných už při návrhu aplikace. Psaní kódu je proto rychlejší, což umožňuje zužitkovat čas a zvýšit dosavadní dovednosti. Uživatelské vývojové prostředí lze přizpůsobit podle potřeb a vkusu každého programátora. Obsahuje sady nabídek, které jo možné pomocí propracovanéhosystému ukotvit jako panel to aplikace na libovolná místa. Vytvořené aplikace lze provozovat na různých platformách, jejichž počet stále roste. Nové nástroje, integrovaná podpora vývoje řízeného testy, jakož i nové ladicí nástroje pomáhají rychle a snadno odhalit chyby, což přispívá k vysoké kvalitě aplikací vyvinutých pomocí software Visual Studio 2010. Díky funkčním frameworkům na straně serveru i klienta je vývojář schopen rychle budovat webové aplikace, integrované s libovolným datovým úložištěm, běžící v kterémkoliv moderním prohlížeči a s plným přístupem k aplikačním službám ASP.NET a k platformě Microsoft.

Předdefinované šablony

Microsoft Visual Studio 2010 obsahuje množství předdefinovaných šablon pro všechny nejdůležitější procesy anebo diagramy, které chcete nakreslit.

Rychlé kreslení
Microsoft Visual Studio 2010 automaticky napovídá další logické tvary, které je možno nakreslit. Rychlost umožňují velké knihovny ikon a nové intuitivní uživatelské rozhraní.

Zjednodušení komplexních diagramů
Složité procesy lze zjednodušit zobrazením podprocesů jako jeden objekt. Detaily se zobrazí kliknutím na ně. Důležité tvary lze zvýraznit a odlišit speciálním tvarem tzv. kontejnerem.

Automatické formátování
Automatické formátování je otázkou několika kliknutí na předdefinované formáty.

Jednoduché sdílení a dynamická data
Diagramy lze sdílet přes webové rozhraní nebo pomocí serveru SharePoint. Diagramy se mohou automaticky aktualizovat a ukazovat stav jednotlivých částí procesu, a to pomocí čárových ukazatelů nebo budíků. [11]

2.6 Programovací jazyk C#

Jazyk, v němž je aplikace napsána se nazývá C# nebo také C Sharp. C# je objektově orientovaný programovací jazyk. Byl vytvořen společností Microsoft společně s platformou .NET a později standardizován organizací ECMA a ISO. Byl založen na jazycích C++ a Java. Je tedy potomkem jazyka C, ze kterého čerpá syntaxi. Je jedním z programovacích jazyků podporující spojení psaného kódu a kódu vytvořeného pomocí uživatelského prostředí. Specifikace jazyka definují prostředí, které umožňuje použití jazyků na vysoké úrovni užívání na různých počítačových platformách, aniž by byl přepsán pro specifické platformy. C# má být moderním jazykem, zaměřeným na objektově orientované jazyky. C# se v dnešní době používá hlavně k tvorbě databázových programů, webových aplikací a webových služeb, formulářových aplikací hlavně pro systémy Windows. Nejnovější verze jazyka je C# 4.0, který byl představen 12. dubna 2010. Cílem jazyka C# je být objektově orientovaný, mnohoúčelový, přitom ale jednoduchý a moderní. Jazyk C# a jeho implementace poskytují podporu při psaní kódu v podobě hlídání hranic polí, hlídání nepoužitých nebo neinicializovaných polí a automatickou úpravu kódu do přehledné podoby, zvláště pro cykly a funkce. Jazyk je vhodný pro softwarové komponenty distribuované v různých prostředích. Důležitá je přenositelnost kódu, zvláště pro programátory, kteří znají C a C++. C# aplikace jsou zaměřeny na minimální požadavky na paměť a výkon při zpracování požadavků: Jazyk přesto není určen k přímému soutěžení o výkon a velikost s jazykem C nebo jazyky symbolických instrukcí.
Při vývoji .NET Framework byly napsány třídy, používající spravovaný kód kompilátorem, zvaný jednoduché spravované C (simple managed C - SMC). V lednu 1999 byl vytvořen tým vedený Andersem Hejlsbergem, který začal budovat jazyk, který byl nazván C jako objektově orientovaný jazyk. V červenci 2000 byl veřejně oznámen na profesionální konferenci vývojářů a byl přejmenován na C#.
Hlavní designérem a vedoucím projektu C# ve společnosti Microsoft se stal Anders Hejlsberg, který se již dříve podílel na konstrukci Turbo Pascalu, Delphi Embarcadero (dříve CodeGear Delphi a

Borland Delphi), a Visual J++. V rozhovorech a technických zprávách uváděl, že: „vady ve většině hlavních programovacích jazyků (např. C++, Java, Delphi a Smalltalk) se řídily základy Common Language Runtime (CLR), který řídil design v jazyku C sharp jazyk sám„. James Gosling, který vytvořil programovací jazyk Java v roce 1994, a Bill Joy, spoluzakladatel Sun Microsystems, tvůrce Javy, nazval C# "imitací" jazyka Java. Gosling dále tvrdil, že „C# je druh Javy se spolehlivostí, produktivitou a bezpečností, která byla smazána“. Klaus Kreft a Angelika Langer (autoři C++ stream book) uvedli v blogu, že "Java a C# jsou téměř totožné programovací jazyky. Nudné opakování, kterému chybí inovace“. „Málokdo bude tvrdit, že Java či C# jsou revoluční programovací jazyky, které změnily způsob, jakým budeme psát programy“ a „C# si půjčil hodně z Javy. A naopak nyní, C# podporuje boxing a unboxing, my budeme mít velmi podobné funkce v jazyce Java„. Anders Hejlsberg tvrdí, že C# není "klon Javy" a je „mnohem blíže k C++“ v jeho designu. [12]

Verze jazyka C#:

C# 1.0 (leden 2002)
První verze, která byla vydána společně s .NET Frameworkem 1.0 a obsahovala základní podporu objektového programování, ve kterém vycházela z jazyka C++. Vývojovým nástrojem byl program Microsoft Visual studio .NET 2002.

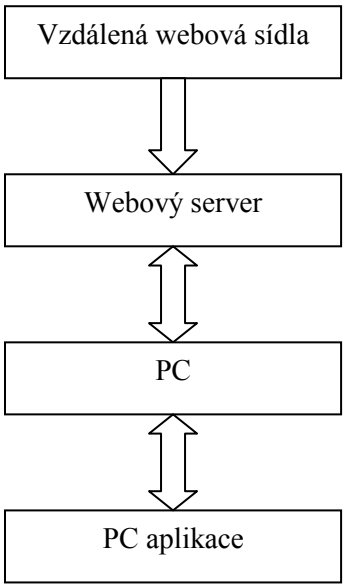
C# 2.0 (listopad 2005)
Další verze jazyka C# vyšla na konci roku 2005. Přinesla nové vlastnosti jako:
Nativní podpora generik vycházející z podpory na úrovni CLI, částečné a statické třídy, iterátory, anonymní metody pro pohodlnější užívání delegátů a nullovatelné hodnotové typy a operátor koalescence. Vývojovým nástrojem byl program Microsoft Visual studio .NET 2005.

C# 3.0 (listopad 2007)
Vyšla v roce 2008 společně s .NET Frameworkem 3.5. Obsahuje poměrně revoluční změny, které však nevyžadují změny podkladového IL, takže aplikace v něm psané půjdou spouštět i na počítačích vybavených toliko druhým Frameworkem.
LINQ (Language Integrated Query) zavedlo několik nových klíčových slov převzatých z jazyka SQL, za pomocí kterých se lze dotazovat na objekty reprezentující databáze, XML soubory, nebo cokoliv dalšího. Vývojovým nástrojem byl program Microsoft Visual studio .NET 2008.

C# 4.0 (duben 2010)
C # 4.0 je nejnovější verze programovacího jazyka C#, který byl spuštěn 11. dubna 2010. Společnost Microsoft vydala také .NET Frameworkem 4. Hlavním cílem C# 4.0 je interoperabilita s částečně nebo plně dynamicky napsanými jazyky a rámcy, jako například Dynamic Language Runtime.
Vývojovým nástrojem je program Microsoft Visual studio .NET 2010.[13]

3 Návrh aplikace pro monitoring vzdálených webových sídel

Vzdálená webová sídla představují měřící stanice, které odesílají změřená data na webový server.
Webový server obsahuje změřené hodnoty z měřících stanic. Vzhledem k počítači se server chová jako server a počítač jako klient, který požaduje data umístěna na serveru. Jedná se o vztah Server-Klient.



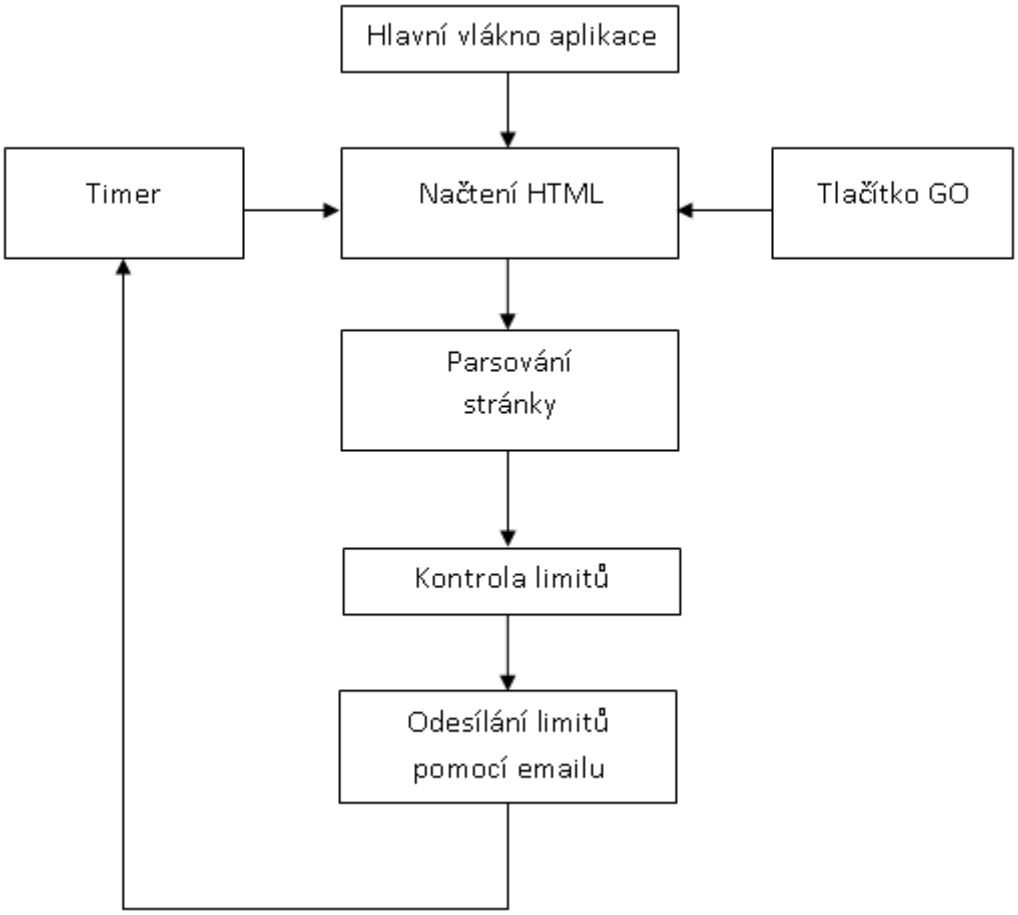
Obr.1: Návrh aplikace

Server-Klient je síťová počítačová architektura, která rozlišuje klienta-aplikaci s grafickým rozhraním a server, kteří mezi sebou komunikují prostřednictvím počítačové sítě. Počítač jako klient požaduje od serveru jeho služby. Na tomto modelu funguje např. email, nebo přístup k databázi. Každý klient může požadovat data od jednoho nebo i více serverů najednou. Servery mohou tyto požadavky akceptovat, zpracovat a vrátit uživateli potřebná data.

Charakterizování klienta: Klient je aktivní část architektury. Jeho úkolem je posílat žádost na server a očekávat jeho odpověď. Obvykle je připojen k omezenému počtu serverů najednou. Klient nemusí znát vnitřní strukturu systému, od něhož data požaduje.

Charakterizování serveru: Server je pasivní část architektury. Jeho úkolem je přijímat požadavky klientů a jejich následné zpracování. Při obdržení požadavku jej v nejkratší možné době zpracuje a odešle klientovi odpověď.

PC je počítač, na kterém je aplikace spuštěna. Vzhledem k aplikaci se chová jako server, ale vzhledem k webovému serveru se chová jako klient. Počítač tedy funguje jako přímý prostředník mezi webovým serverem, na kterém jsou umístěna data a PC aplikací, která data požaduje.[14]



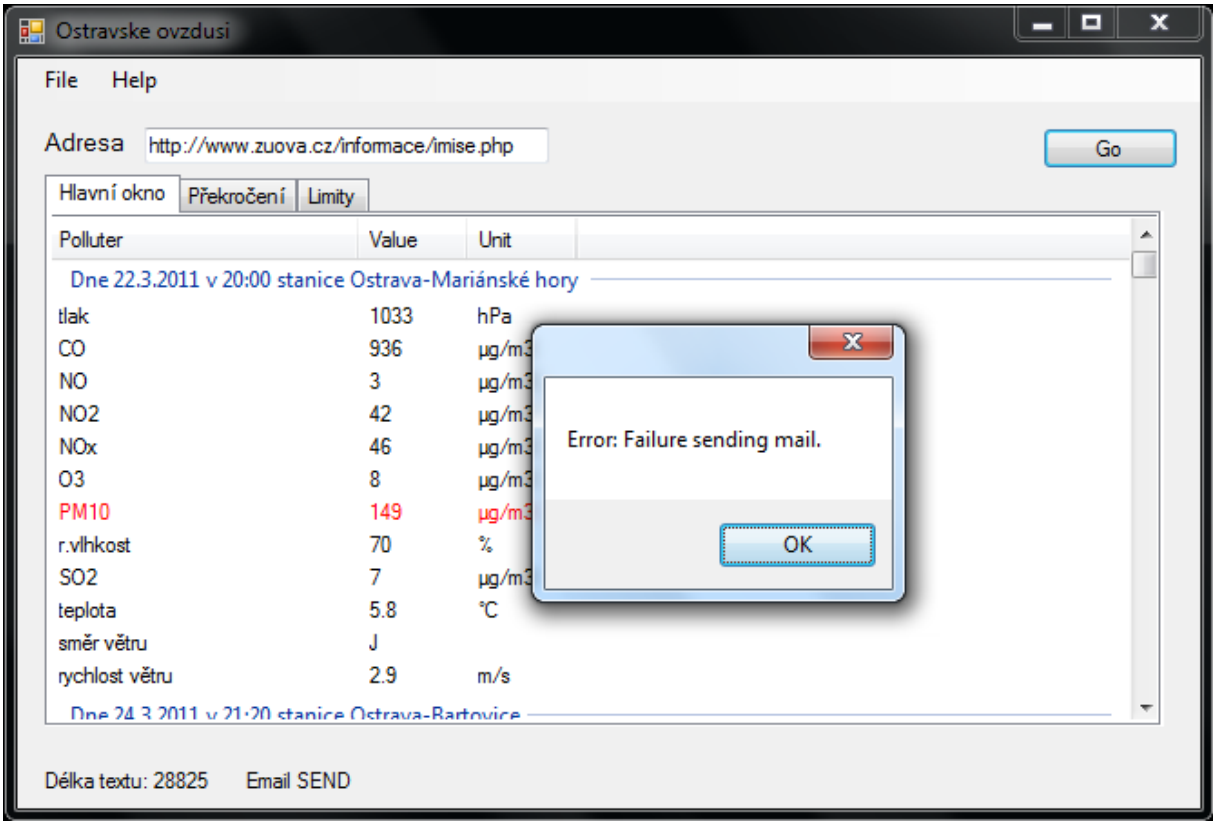
Obr.2: Návrh struktury aplikace

PC aplikace je vlastní program, spuštěný uživatelem. Návrh nového řešení programu spočívá v načtení HTML kódu stránky, aby s ním bylo možno pracovat. Samotné načtení kódu proběhne v metodě btn_address_Click, která je vyvolána stiskem tlačítka Go v okně programu, nebo časovačem pro informovanost uživatele v daných časových intervalech. Po kliknutí na tlačítko metoda načte z textového pole adresu zvolené stránky a předá ji jako parametr s použitím knihovny System.Net. Pomocí instance třídy StreamReader je načítán kód stránky s metodou ReadToEnd() od začátku do konce. Tento kód je následně zobrazen v textovém poli. Pro informaci je zobrazen v dolní části programu délka načteného kódu.

Parsování stránky tedy úprava kódu proběhne ihned po jejím načtení. Aby bylo možné zpracovávat daný obsah stránky, je nutné vědět, jak jsou programově definovány položky v kódu. Toto je pro každou stránku jiné. Z tohoto důvodu je těžké vytvořit univerzální kód použitelný na všechny stránky. Tímto problémem se zabývají HTML parsery, které mají specifikovány tagy stránek a kód stránky rozparsují na části a ty výkonnější je upraví do podoby XML, které je mnohem více srozumitelné. Pro tuto aplikaci nebyl nalezen vhodný parser, který by stránku přijatelně upravil. Řešení je vytvoření vlastního parseru, který projíždí kód stránky jako text a zaznamenává si poslední načtené znaky. Tyto znaky jsou porovnány s uvozovacím tagem. Při shodě dojde k aktivaci pomocného řetězce do něhož jsou vypisovány všechny nalezené znaky. Tento zápis probíhá do

nalezení ukončovacího tagu. Při tomto nálezu se do pomocného řetězce přestávají přidávat načítané znaky, ale čtení kódu stránky pokračuje dále do dosažení další části pro výpis, kdy je načten její obsah, nebo do konce kódu stránky. Dosažení konce stránky je indikováno programem, kdy ukončovací znak je shodný s délkou kódu stránky.

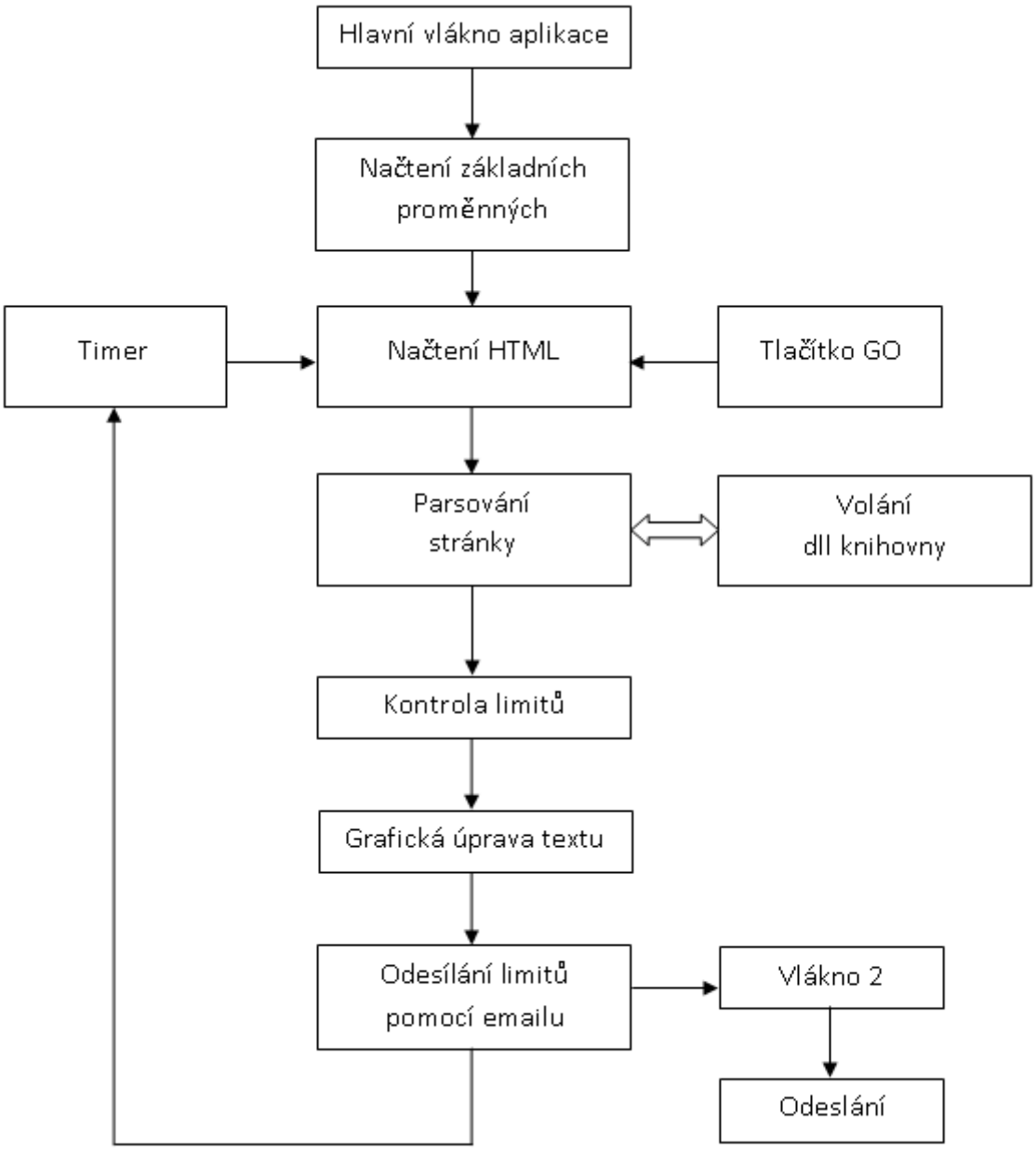
Po dosažení konce je vypsán výsledek parseru z pomocného řetězce. Výsledek parseru se porovná s limity v ovzduší a je očekáván email, na který po jeho zadání budou odeslána data, která jsou navíc vypsána v textovém poli. Pro odesílání emailu je použito knihoven System.Net a System.Net.Mail. Třída MailMessage a konstruktor MailMessage() vytvoří instanci pro nový email. Pomocí několika metod je novému objektu Msg přiřazeno několik parametrů nutný pro odeslání emailu. Těmito parametry jsou odesílatel, příjemce, předmět, text zprávy a smtp server, který zprávu odesílá. Aplikace na správnou operaci nereaguje, při vzniku chyby aplikace informuje uživatele zprávou s výpisem chyby.



Obr.3: Chyba při odesílání emailu

3.1 Struktura aplikace

Struktura aplikace je velmi podobná jejímu návrhu a přímo z něj vychází. Hlavní vlákno aplikace zpracovává svůj kód sekvenčně. Nastalá odchylka od sekvenčnosti příkazů by způsobila nefunkčnost aplikace. Sekvenčnost přímo souvisí s požadavky na funkci aplikace.



Obr.4: Struktura PC aplikace

Nejdříve je nutné dostat zadání, poté zpracovat ve výsledky a řešení odeslat. Zadáním pro danou sekvenci se stává v načtení kódu, zpracování zadání v úpravě textu a odeslání výsledků v podobě odesílání emailu a grafickém rozhraní aplikace-uživatel. Výsledná struktura programu je až na připojené části téměř totožná.

Při startu aplikace jsou načteny základní proměnné, jako uživatelské nastavení nebo definované hodnoty bezprostředně nutné pro správný chod aplikace. Načtení HTML kódu probíhá ihned po stisknutí akčního tlačítka nebo příkazem timeru.

Ukončením načítání HTML do aplikace je uveden v činnost parser aplikace. Načítání kódu bylo v jazyku C# zrušeno a nahrazeno pomocnou DLL knihovnou v jazyku C. Toto řešení poskytlo nebývalou rychlost a výrazně tak zrychlilo celkový chod aplikace, což má za následek snížení výkonových nároků na počítač a možnost širšího využití aplikace na jiných počítačích. Tímto se také odstranil jeden z hlavních problémů aplikace a to tím, že aplikace zamrzávala na obrazovce, kdy se čekalo, až parser dokončí svou práci a bude zobrazen zpracovaný kód na hlavním okně aplikace. Tato část zpracovávaného úkolu nemůže být zpracována jiným vláknem. Jiné vlákno by aplikaci umožnilo paralelní běh, ale porušila by se sekvenčnost příkazů. Vlákno zpracovávající načítání by skončilo později než hlavní vlákno, což by mělo za následek, že daší funkce aplikace by nedostaly žádná data a výsledek aplikace by byl nulový. Tato část zůstala pro jednoduchost v činnosti hlavního vlákna aplikace.

Kontrola limitů probíhá podle návrhu aplikace. Samotná kontrola probíhá v jazyce C#. Zpracovávaný kód je mnohem kratší než kód, který používá parser, proto zpracování bylo ponecháno v jazyce C#. Tato část kódu je zpracována taktéž sekvenčně, neboť na ní přímo navazuje grafická úprava výsledného textu. Doba zpracování této části není kritická, je mnohem kratší než předchozí kroky aplikace a uživatel již má nějaká data zobrazená v okně aplikace. Tato část kódu byla vhodně upravena do podoby datového pole, které je použito pro grafické zpracování aplikace. Výsledné překročené limity jsou zobrazeny v aplikaci v záložce Překročení. Tyto limity jsou upraveny do podoby seznamu. K tomuto seznamu bylo použito prvku listView.

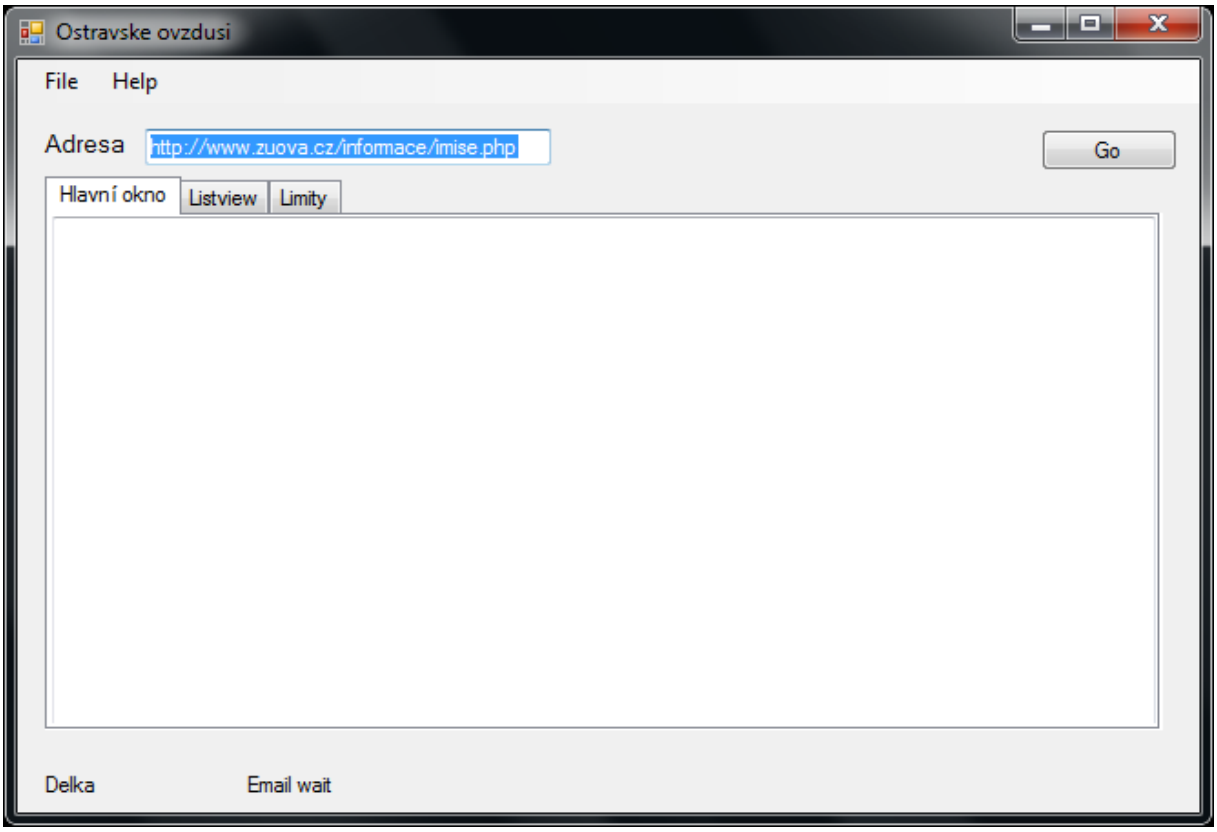
Grafická úprava textu spočívá v grafickém oddělení jednotlivých prvků v textu. V aplikaci je použito zvýraznění textu, konkrétně k vyznačení překročení limitů. Grafická úprava textu poskytuje uživateli velmi dobrý přehled o překročených limitech, než jen holý text. Poskytuje jednodušší text, neboť černý text používaný např. v textBoxu nelze v průběhu psaní změnit. Možnost upozornění na části textu jsou tak velmi omezené. Text přepsán do listView je možno odlišit již velice jednoduše a to změnou uspořádání, nebo i změnou barvy textu, které aplikace využívá. Výhodou i nevýhodou listView je uspořádání textu do struktury pole. Každý prvek může být popsán jednotlivými parametry, ale musíme do celého textu zobrazovaného uživateli adresovat každý prvek zvlášť. Zvýraznění textu tedy přináší uživateli dobrý přehled za cenu složitější aplikace.

Po ukončení činnosti grafické části jsou limity odeslány pomocí emailu. Odesílání emailu probíhá nízkou rychlostí, zapříčiněnou složitostí tvorby a komunikace aplikace s SMTP klientem, přes který se email posílá a odesíláním dat na cílovou schránku. Tato část kódu již nepodléhá přísné sekvenčnosti, neboť zpracování emailu není vázáno na stav aplikace. Zde je vytvořeno pomocné druhé vlákno, které zajistí odeslání emailu z aplikace. Po odeslání je vlákno smazáno.

3.2 Hlavní okno aplikace

Hlavní okno aplikace slouží k přímému ovládání aplikace Ostravske ovzdusi. Při startu aplikace je uživateli dána přednastavená aplikace s defaultními hodnotami, schopná samostatného běhu již po

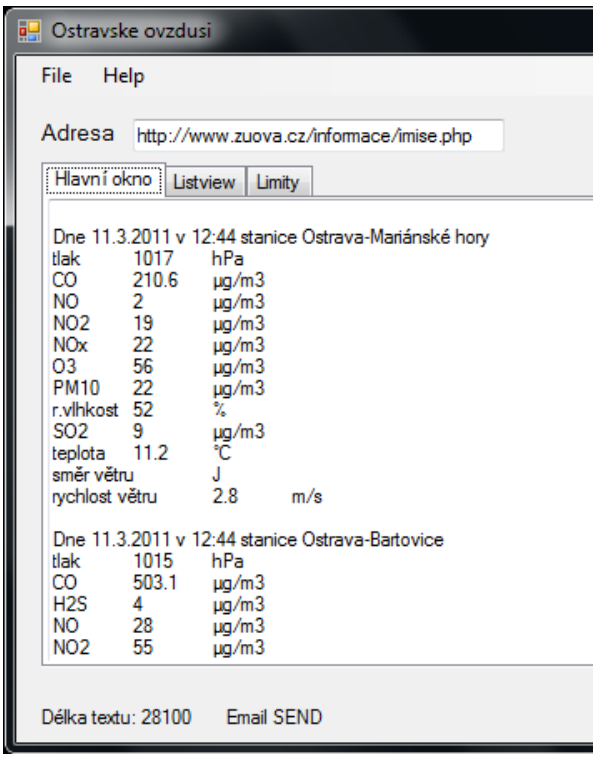
stisknutí jednoho tlačítka. Po vzhlednové stránce vévodí oknu velké textové pole, do nějž jsou zapisovány veškeré údaje právě prováděného procesu programu. Nad velkým textovým polem se nachází pole pro zadání adresy webové stránky, na které se nacházejí změřená data z měřících stanic. Defaultně jsou nastaveny ostravské stanice v Mariánských horách, Bartovicích, Přívoze, Mizerově a Fifejdách. V těchto stanicích dochází k pravidelnému měření emisí a škodlivin ve vzduchu jako jsou oxid uhelnatý, oxid siřičitý, oxid dusičitý, prach o velikosti částic 1 2,5 a 10um a další.



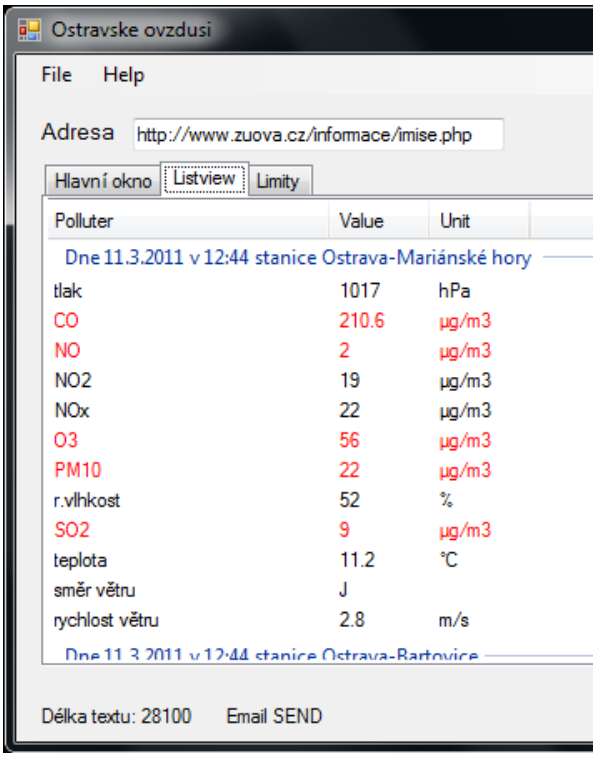
Obr.5: Hlavní okno

Start aplikace je spuštěn samotným startem aplikace, po jejímž spuštění defaultně nastavený časovač na jednu hodinu, se automaticky pokusí získat data z měřících stanic a start aplikace je taktéž umožněn uživateli stisknutím tlačítka Go , po jehož stisknutí se provedou stejné operace jako při akci časovače. Uživatel při prvním platném spuštění aplikace je aplikací obeznámen o stavu škodlivin v ovzduší. Uživatel, jenž doposud ponechal defaultní nastavení je informován o stavu aplikací po dobu jedné hodiny od spuštění aplikace. Nastavení aplikace je provedeno v nabídce File / Settings, kde je uživateli umožněno změnit defaultní nastavení aplikace. Po provedení změn je časovač nastaven na hodnotu zvolenou uživatelem a při dalším plánovaném spuštění je aplikace spuštěna v tuto dobu. Změna časovače se projeví i při vypnutí aplikace, která si změněný časovač zapamatovala. Při zadání emailu aplikace přestává uživatele informovat zprávou, data o překročení škodlivin jsou odesílána na zvolený email automaticky, s výjimkou nastalé chyby, o které je uživatel aplikací informován.

Porovnání textBoxu a listView



Obr.6 : Výsledek aplikace pomocí textBoxu



Obr.7: Výsledek aplikace pomocí listView

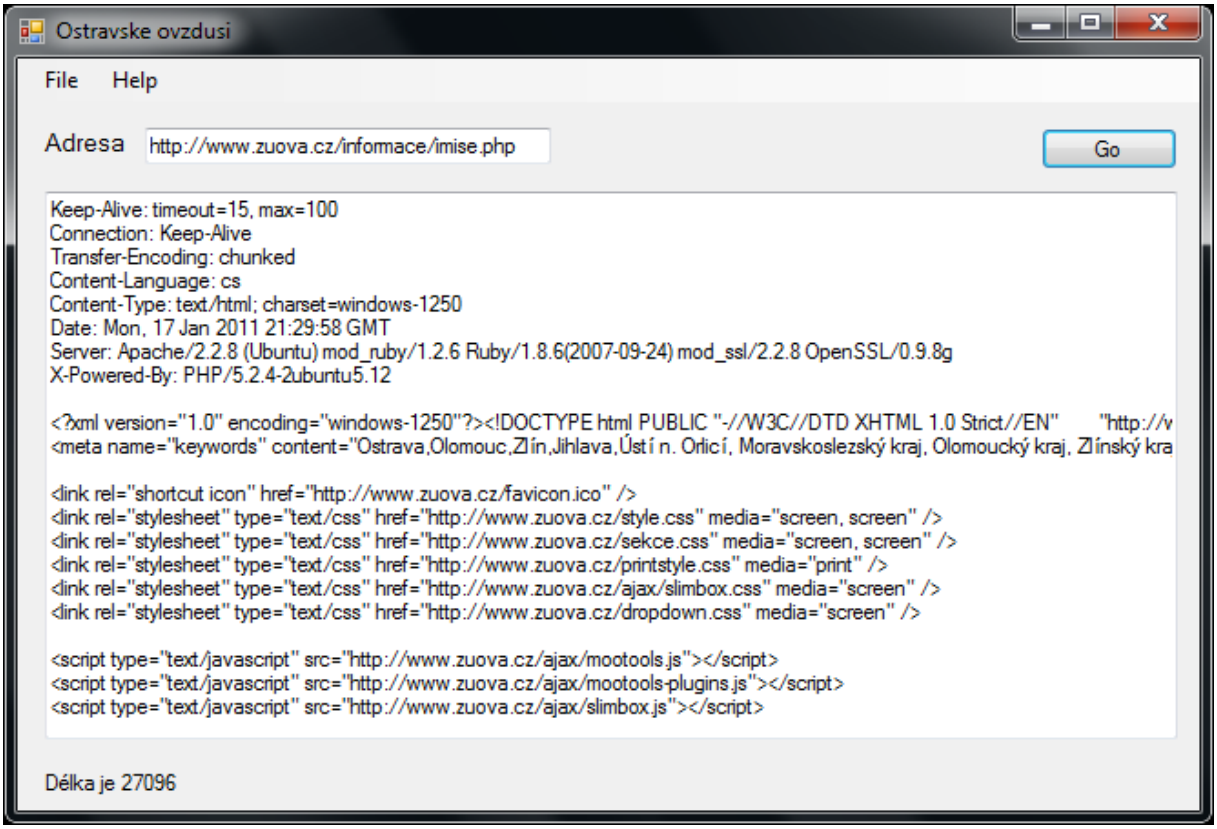
TextBox je jednoduchá datová struktura, do níž je možné vkládat text a používat ji jako uživatelský zdroj informací nebo jako datový výsledek. Umožňuje vkládat uživateli data. V zálkadním nastavení je textbox nastavený na jednořádkový. Textbox lze nastavit na hodnotu Multiline, což znamená, že textbox bude víceřádkový, umožní tedy zobrazit větší část informací. Hodnotou multiline se taktéž stává multivstupní, je možné vkládat více dat v jeden okamžik jako vstup. Zobrazená data lze nastavit jako výstup bez možnosti úpravy zobrazených dat vlastností ReadOnly. Textbox lze graficky upravit, ale pouze v celém svém rozsahu. Umožní změnu barvy pozadí vlastností BackColor a to i při běhu programu. Změna barvy textu je možná metodou ForeColor. Umožní změnu velikosti a stylu písma. Pro dynamickou změnu okna lze ”přichytit“ textbox k okrajům okna a umožnit dynamickou změnu velikosti, podle velikosti okna, až do maximálních parametrů, defaultně neomezených.

ListView je složitější datová struktura umožňující datové třídění prvků a jejich značení. Listview umožňuje text pouze zobrazit, neslouží jako vstup dat. Je tedy vhodný jako datový výsledek. Podobně jako textbox umožňuje listview úpravu textu v podobě barvy textu a pozadí. Oproti textboxu však listview sdružuje text na bloky, což umožní změnu barvy, velikosti, pozadí jen pro konkrétní prvek, bez omezení dalších. Toto je vhodné např. pro zvýraznění určitého textu, na který je kladen větší důraz na pozornost uživatele. Příkladem v aplikaci jsou zvýrazněné nadlimitní hodnoty škodlivin v ovzduší. Pro porovnání stejné hodnoty na obr.5 s textboxem a na obr.6 s použitím listview. Z obrázků je patrný velký rozdíl, mezi obyčejným textem textboxu a složitějším listview. Zatímco grafickou úpravu textu v textboxu bylo možné provézt pouze pomocí programovacích značek jako tabelátor, použitím

listview bylo dosaženo přehledné struktury, která umožnila odlišení důležitých částí jako překročení limitů. Listview umožnilo také seskupit data do skupin, v tomto případě do skupin podle místa měření, na obr.6 podle skupiny Ostrava-Mariánské hory. Listview umožní uživateli vložit obrázek na pozadí, což učiní z holého místa na ploše aplikace zajímavější, graficky lépe vypadající prostředí pro uživatele. Listview je tedy určen pouze jako datový výstup s možností grafické úpravy. Oproti textboxu, který může sloužit jako vstup i výstup je listview možno lépe graficky propracovat, ovšem za cenu větší složitosti psaní kódu.

3.3 HTML načtení kódu

Spojení aplikace a webové stránky probíhá pomocí funkce address, jejímž výsledkem je do textového pole v hlavním okně zapsán html kód samotné webové stránky. Načítání kódu do aplikace je jištěno podmínkou try a catch, která zabezpečuje, že při nesprávném dotazu na webový server nedojde k havárii aplikace nebo systému, na kterém aplikace běží. Dotaz na server ve formě request je serverem vrácena do formy response a předána hodnotě responsetext zadefinovanou jako text. Odpovědí serveru je tedy celistvý text, zobrazen do textovéh pole aplikace. Po zpracování tohoto požadavku jsou volány funkce pro úpravu načteného textu.



Obr.8: HTML kód načtený aplikací

3.4 HTML parser

Funkce parser

První z funkcí, která je volána po načtení samotného html kódu do aplikace je funkce parser. Funkce parser provede operaci zvanou syntaktická analýza (slangově též *parsování* nebo *parsing*).Je to proces analýzy posloupnosti prvků, jak zjistit jejich gramatickou strukturu vůči dané formální gramatice. Parsováním se vstupní text naformátuje z textového pole do datové struktury pole, jenž je vhodný pro pozdější zpracování dalšími funkcemi a který zachovává hierarchii vstupních dat. Prvním krokem funkce parser je analýza, při níž se ze vstupního textu v textovém poli vytváří posloupnost tzv. *tokenů*, tedy elementárních nositelů výrazu v rámci daného jazyka. Tokenem je např. závorka, znak, číslo, řetězec, sada znaků, symboly a pod.

Původní HTML parser napsán v jazyku C# byl nahrazen obdobným kódem přepsáním do jazyka C. Řešení v jazyce C přineslo do aplikace značné zrychlení v podobě zpracování parsovaného textu. Původní kód napsaný v jazyce C# bylo možné mým počítačem zpracovat za přibližně 91369 ms. Nově napsaný parser v jazyce C zvládne tentýž počítač zpracovat za 0,5 ms. Z důvodu příliž malého času se měření provedlo 200x a čas poté zprůměroval. Z naměřených časů je znát, že provádění stejného úkonu v jazyce C bylo 182738x rychlejší než v jazyce C#. Délka zpracovávaného kódu se pohybuje okolo 30 000 znaků. S rostoucí délkou textu se čas zpracování násobně zvyšuje. K měření času bylo využito funkce Enviroment.Tickcount, která měří čas od zapnutí PC v ms.
HTML parser je napsán jako funkce. Jelikož není možné zařadit kód v jazyce C do jazyka C# je nutné tyto dva jazyky spojit přes rozhraní, kterému oba dané jazyky rozumí. Jako rozhraní bylo využito DLL knihovny, se kterou umí oba jazyky komunikovat.

DLL

Co je DLL

DLL (**Dynamic-link library**, dynamicky linkovaná knihovna)
DLL je knihovna obsahující kód, z níž mohou být data používány více než jedním programem současně. Každý program proto můžete použít funkce obsažené v knihovně DLL k implementaci vlastní operace. To pomáhá zvýšit opětovné použití kódu a efektivní využití paměti. Pomocí DLL program může být rozporcován na samostatné funkční celky. Například účetní program může být prodán o částech. Každou část lze načíst do hlavního programu při spuštění, pokud je část nainstalována. Protože části jsou samostatné celky, vytížení procesoru programem je menší a části jsou načítány pouze v případě, že je jich požadováno. Aktualizace jsou navíc snazší, použití pro každou část bez ovlivnění ostatních částí programem. Výše mezd a sazeb daní se mění každý rok. Tyto změny jsou izolované v knihovnách DLL, mohou se použít aktualizace bez nutnosti reinstalace celého programu.
Následující seznam popisuje některé soubory, které jsou implementovány jako knihovny DLL v operačních systémech Windows:

- **Soubory prvky ActiveX (OCX)**
  Příklad ovládacího prvku ActiveX je ovládací prvek Kalendář, který umožňuje vybrat datum z kalendáře.
- **Soubory ovládacích panelů (CPL)**
  Příklad souboru CPL je položka, která je umístěna v Ovládacích panelech. Každá položka je specializovaný DLL.
- **Soubory ovladače (.drv) zařízení**
  Ovladač tiskárny, který řídí tisk na tiskárně je například ovladač zařízení.

Výhody DLL

- **Využívá méně prostředků**
  Použití funkce v knihovně DLL může snížit duplikaci kódu pro více programů, které jsou načteny z disku a fyzické paměti. To může výrazně ovlivnit výkonu nejen programu, který je spuštěn v popředí, ale také dalších programů, které jsou spuštěny v operačním systému.
- **Modulární architektura Promotes**
  DLL pomáhá zvýšit vývoj modulární programů. To pomáhá vyvinout velké programy, které vyžadují více jazykových verzí, nebo program vyžadující modulární architekturu. Například modulárním program je účetní program, který má mnoho částí, které lze dynamicky načíst při spuštění.
- **Usnadňuje zavedení a instalace**
  Pokud funkce v rámci DLL potřebuje aktualizaci nebo opravu, zavedení a instalace DLL nevyžaduje, aby se knihovna DLL znovu připojila k programu. Pokud více programů používá stejné DLL, více programů bude využívat všechny výhody aktualizace nebo opravy. K tomuto problému může dojít častěji při použití DLL, která je pravidelně aktualizována nebo opravována.

DLL závislosti

Když program nebo DLL používá funkce v jiném DLL, je vytvořena závislost. Proto program již není soběstačný a může docházet k problémům, pokud je přerušena závislost. Program například nemusí pracovat, pokud dojde k jedné z následujících akcí:

- Závislá DLL byla inovována na novou verzi
- Závislé DLL je opravena
- Dřívější verze je přepsána závislou DLL
- Závislá DLL je odebrána z počítače.

Tyto akce jsou obecně označovány jako DLL konflikty. Pokud není vynucena zpětná kompatibilita, nemusí se úspěšně spustit program.[15]

Problematika DLL

Původní koncept práce předpokládal, že program bude napsát celý v jazyce C#. Jazyku C# bylo zvoleno z důvodu snadné tvorby grafické části programu, tvořené ve visual studiu. Toto řešení se s přibývající složitostí programu začalo projevovat výraznými prodlevami mezi odezvou aplikace a příkazy užiivatele. Aplikace začala „zamrzávat“ a bylo nutné tento stav eliminovat.
Jedním z možných řešení by bylo použití vláken. Pomocí vláken by se aplikace z tohoto stavu dostávala pomocí jiného vlákna, které by na krátkou chvíli umožnilo činnost jiné části kódu a uživatel by mohl s aplikací lépe komunikovat. Použití vláken by mělo pozitivní vliv na uživatelské prostředí, ale mělo by negativní důsledky na rychlost zpracování dat aplikací. Použitím jiného vlákna, které by umožnilo komunikaci aplikace a uživatele by neumožnilo zpracovávat aktuálně prováděná data, čímž se doba zpracování prodlužuje o dobu, kdy pracuje jiné vlákno, než to, které zpracovává data.
Možnost použití vláken by tedy nepřineslo do aplikace mnoho výhod.

S narůstající složitostí programu se prodlužovala i doba, po kterou byly data aplikací zpracovávána. Bylo tedy nutné najít nové řešení, takové které by urychlilo běh aplikace. V úvahu připadala možnost přepracování algorytmu pro zpracování dat s použitím rychlejších funkcí a požití vláken jako částí pro jejich zpracování. Tato možnost byla také zamítnuta, protože k výraznému zrychlení zpracování dat opět nedošlo. Konečnou možností, která byla realizována bylo použití DLL knihovny. Tato možnost připadla do úvahy při testování rychlosti funkcí, kdy bylo zjištěno, že použití jazyka C v DLL knihovně jako externí funkce přinesla výrazné zrychlení. Pouhým přepsáním algorytmu zpracovávající data z jazyka C# do jazyka C bylo dosaženo velmi uspokojivých výsledků. Obavy o rychlost komunikace mezi aplikací a DLL knihovnou byly zbytečné. Výměna dat probíhá velmi rychle. Při použití DLL knihovny byl čas původního algorytmu 91369 ms. Při použití DLL knihovny byl čas zpracování od poslání dat do jejich přijetí přibližně 0,5 ms. Tento algoritmus byl otestován na jiném PC, zde ale změna nebyla tolik výrazná. Původní algoritmus trval přibližně 21 vteřin a přepsaný algorytmus trval 15ms.

Testování rychlosti

Pro otestování rychlosti zpracování dat aplikací byla vytvořena aplikace, která měří čas zpracování úlohy. Nová aplikace zpracováná text, podobně jako aplikace, pro kterou byla navrhnuta. Cílem je ukázat jak rychle různé počítače zpracují danou úlohu. Úkolem aplikace je načíst vstupní text a nahradit určité znaky. Pro přesnější měření je každý cyklus spuštěn 150 krát za sebou. Tento text je zpracován pomocí jazyka C# a DLL knihovny s použitím jazyka C. Výsledkem aplikace jsou časy, za které je tento text zpracován. Vstupní text má 31888 znaků a jde o HTML kód.

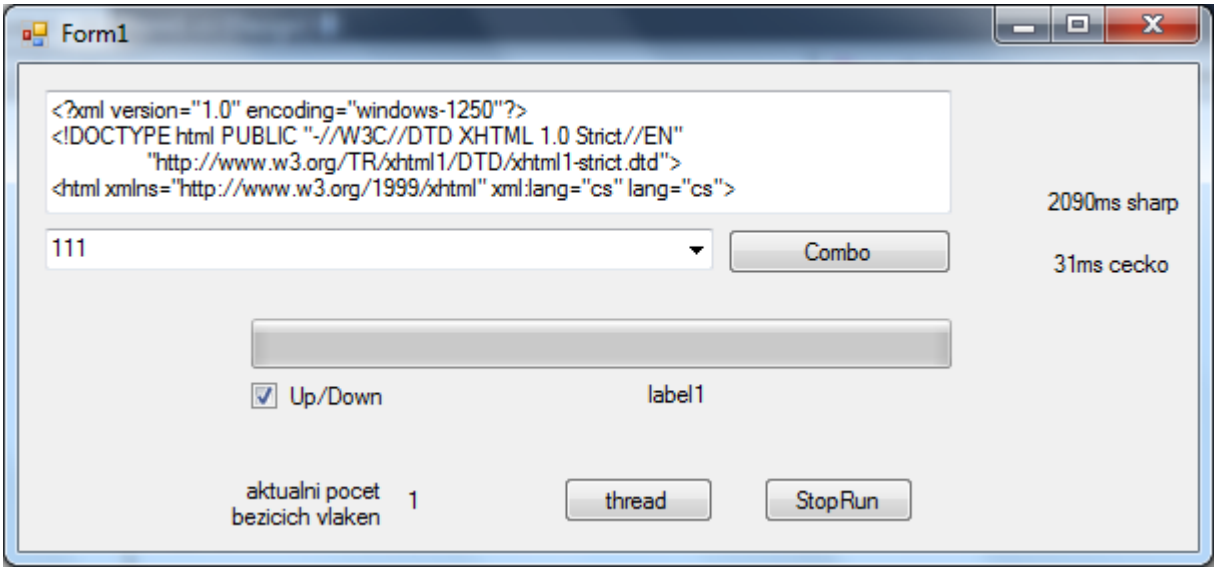
Výsledky:

Můj notebook:
procesor: Intel P7350 , 2GHz
rychlost disku: 5400 RPM
windows: 7 Professional
C# = 2090 ms

C= 31 ms
Zvýšení rychlosti = 67x

PC2: Stolní PC
procesor: AMD Sempron, 1,6GHz
rychlost disku: 7200 RPM
windows: XP Professional
C# = 2797 ms
C= 63 ms
Zvýšení rychlosti = 44x

PC3: HP ProBook 4520s W7
procesor: Intel i3 380M, 2,53GHz
rychlost disku: 7200 RPM
windows: 7 Home Premium
C# = 1787 ms
C= 140 ms
Zvýšení rychlosti = 13x

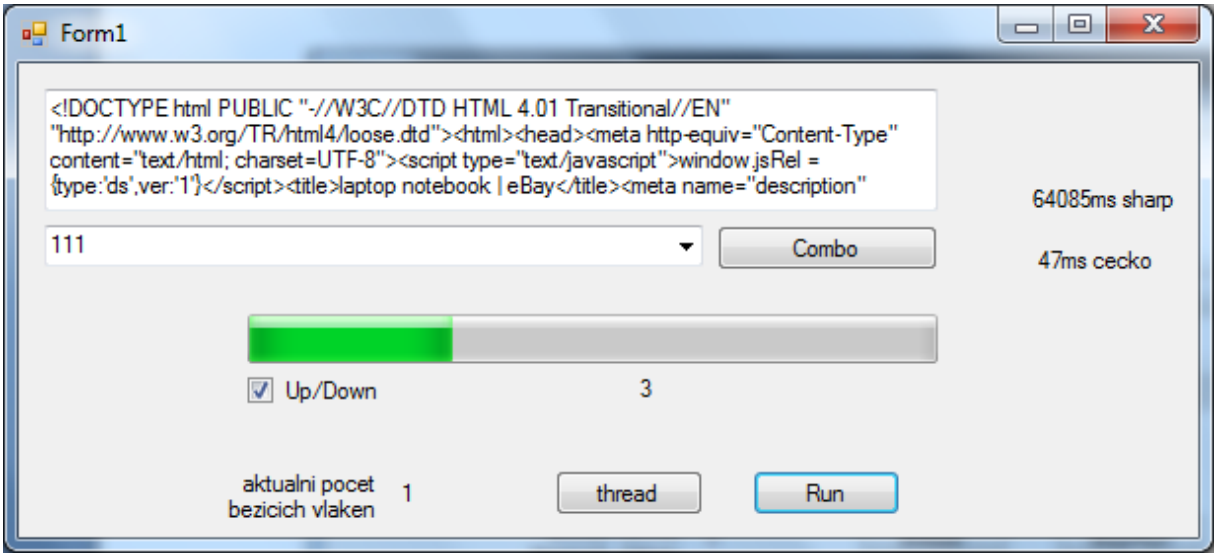


Obr. 9: Test rychlosti

Algorytmus zpracovávající text v jazyce C# se jeví vždy jako nevýhodný. Zpracovávání pomocí DLL knihovny v jazyce C bylo vždy znatelně rychlejší. Tento jednoduchý test prokázal, že použití DLL knihovny značně urychlí běh aplikace. Pro výraznější ukázku síly výpočetního výkonu byl vytvořen složitější algorytmus, kterému byl zadán text v délce 491576 znaků a zadáno prohledávat a zpracovat více znaků a jejich změna. Předpokladem bylo větší rozdíl mezi jazykem C# a jazykem C. Zde se již ale jazyk C# dostává do mírné výdody, protože data má přímo v programu, jazyk C je musí teprve načíst z aplikace. Tento jev nelze eliminovat, protože měření času probíhá v aplikaci. Měření má

ukázat jak rychle zpracuje text jazyk C, je nutné tedy počítat i s časem, který je nutný k přesunu dat mezi aplikací a DLL knihovnou a zpět.

Obr.10 ukazuje rozdíl mezi jazykem C# a jazykem C s použitím složitějšího algorytmu.



Obr. 10: Test rychlosti 2

Algorytmus 2 dokázal snížit čas z 64085 milisekund na 47 milisekund. To znamená, že jazykem C byl text zpracován 1364 krát rychleji. Algorytmy v jazyce C# a jazyce C jsou totožné.

Činnost parseru v aplikaci

HTML parser zpracovává vstupní text, předaný pomocí jazyka C# do DLL knihovny přímo z aplikace. Vstupní text, jako celek, je poté zpracováván jako řetezec znaků, znak po znaku. Zpracování tohoro druhu tedy klade značné nároky na výpočetní výkon počítače. Text, který je aktuálně zpracováván, se nikde nevypisuje. Určení aktuální polohy zpracovávaného znaku nepřinese uživateli žádné dodatečné informace a běh aplikace by byl razantně pomalejší. Cílem přepsání do jazyka C a použití DLL knihovny by potom přišlo nazmar. Výsledek parseru se exportuje z DLL knihovny do původní aplikace. Výsledkem parseru je opět text, nyní již několikanásobně kratší a tudíž lépe zpracovatelný. Výhodou exportovaného textu je známá struktura, předpřipravená pro další úpravy jako řádkování, značení nebo mazání, která je součástí procedury parseru.

HTML parser použitý pro aplikaci vyhledává sérii až pěti znaků v načteném html kódu stránky. Pro procházení kódu je zvolen cyklus while, který načítá znaky od 0.prvku v textovém řetezci až do posledního. V rámci cyklu while jsou tyto znaky (p, q, r, s, t) naplňovány a jejich kombinace je porovnávána v podmínkách, odpovídajících hledanému výrazu.

Příklad hledaného výrazu:

```
if (d == 0)
{
        t = codesource[i];
```

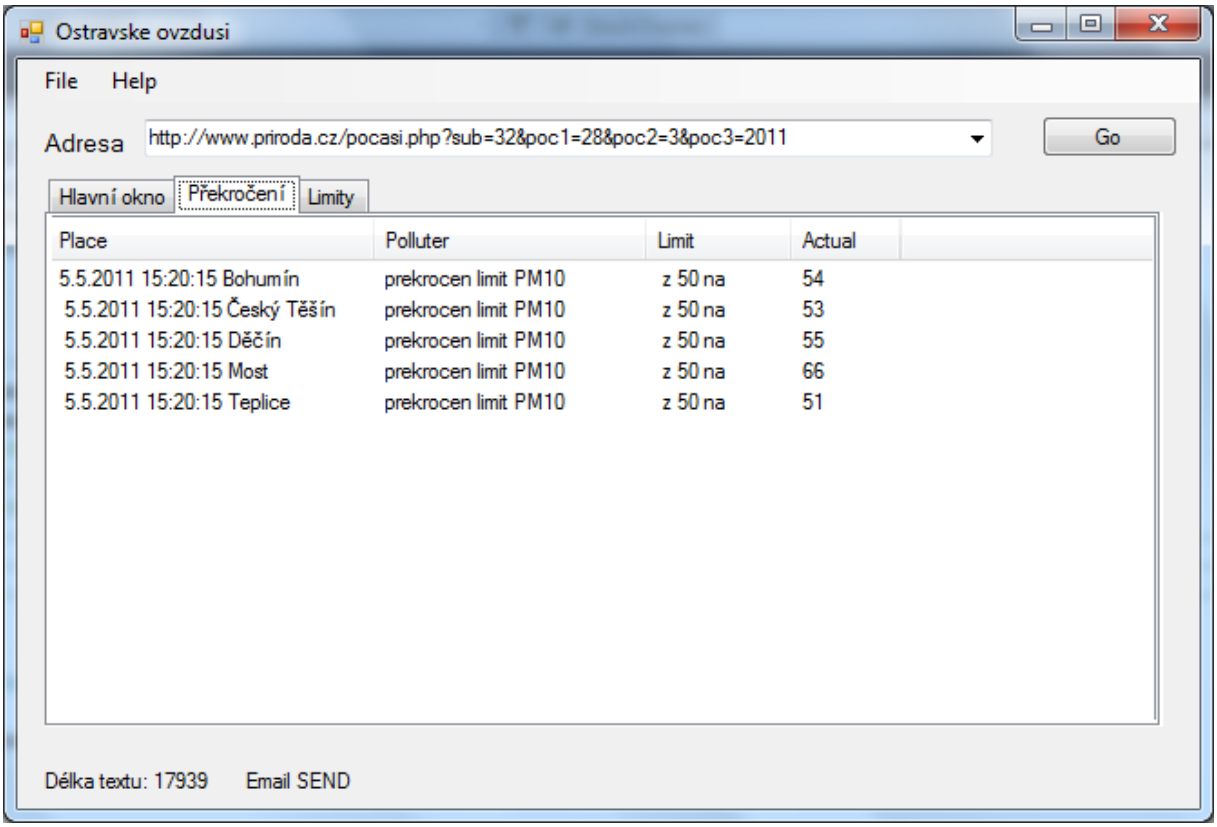
```
 if (p == '<' && q == 'h' && r == '3' && s == '>' )
 {
   d = 1;
 }
if (ci > 0)
{
    newline();
     newline();
}

 if (p == '<' && q == 't' && r == 'd' && s == '>' )
 {
   d = 1;
}

 if (p == 'o' && q == 'n' && r == 'g' && s == '>' )
 {
     d = 1;
}
```

Tyto výrazy byly nalezeny jako společný výraz, předcházející znaky, které nesou informaci. Výrazy jsou rozděleny do tří úrovní. Nejvyšší skupinou výrazů jsou výrazy nesoucí informaci o poloze. Tyto výrazy předcházejí všem ostatním a jsou v HTML kódu vždy obsaženy na určitých místech, na nimiž následují výrazy nesoucí infomace o místě. Tyto výrazy jsou čísla udávající aktuální stav v daném místě. Jsou to důležité parametry. Tyto parametry jsou seřazeny podle předem dané struktury (tabulka) nebo jsou řazeny dynamicky, což znamená, že se mohou v průběhu času objevovat a mizet. Jsou-li řazeny dynamicky, algorytmus je řadí podle značek, které jim předcházejí. V takovém případě je nutné, aby kód obsahoval takové značky, aby bylo možné přiřadit čísla k parametrům. Příkladem je první server zuova.cz, kde dochází k dynamickému řazení hodnot. Algorytmus byl prověřen, když na stránce byla vyjmuta měřící stanice a byla nahrazena jinou. Ostatní servery, které aplikace zvládá, jsou psány tabulkovým stylem. U těchto stránek se předpokládá, že hodnoty nemohou zmizet. Toto řazení má pevně danou strukturu, která udává, na které pozici je jaká hodnota. Na těchto serverech je načteno místo měřící stanice a číselná data. Informace o datech je přidávána podle předem dané struktury stránky. Nejnižší skupinou výrazů zde představují informace o datech z měřící stanice.

3.5 Funkce check

Funkce check je funkce, která je automaticky spuštěna ihned po ukončení parsování. Jejím úkolem je zjistit, zda nebyla překročena nějaká povolená hranice škodlivin ve vzduchu. Vstupní text po zpracování parserem je naformátován na řádky a zapsán v textovém poli textBox1. Text je rozdělen na řádky do pole stringů. Zde se také nachází přesná definice výrazů, které se v textu vyhledávají. Pole stringů je poté podle těchto hodnot prohledáváno. Při nalezení shody je toto zaznamenáno a podstoupeno dalšímu testování.

Přinalezení shody na pozici pole[n] je testována další pozice pole[n+1] na číselnou hodnotu. Limitní hodnoty jsou definovány při startu aplikace. Při překročení hodnoty je stav zaznamenán do pomocného stringu, který slouží jako zdroj překročených hodnot. Testování poté pokračuje do konce pole. Text získaný do pomocného řetězce je také zdrojem textu pro email a je překopírován do pomocného stringu Emailtext. Text získaný překračováním povolených hodnot je dále graficky upraven. K tomuto účelu slouží prvek listview, ve kterém jsou data zpracována. Vstupní text je rozdělen na řadu stringů, které jsou poté zarovnány do buněk na řádku. Takto upravený text je zobrazen v záložce překročení.



Obr.11: Výsledek funkce check

3.6 Funkce pretty

Po ukončení funkce check je spuštěna funkce pretty. Tato funkce upraví text v textovém poli textBox1 do přijatelné grafické podoby a také označí překročené hodnoty. Text v textovém poli je rozdělen do pole stringů. Takto upravený text je formátován do řádku a zarovnán. Text je rozdělen na skupiny, se jménem měřící stanice jako název skupiny a časem, kdy došlo k měření. Čas měření je shodný s měřící stanicí, která zapisuje, kdy došlo k měření, nebo v případě nenalezení času je čas shodný s aktuálním. Poté jsou vypsány data z měřící stanice. Řádky jsou zkontolovány a v případě, že došlo k překročení, je tento řádek zbarven červeně. Takto lze snadno v textu odlišit mezi daty překročené hodnoty.

3.7 Zasílání výsledků emailem

Zasílání výsledků je prováděno pomocí funkce emailme, která pracuje ve vlákně. Použití vlákna bylo zvoleno z důvodu dlouhé odezvy mezi odesláním a zpětné komunikace aplikace s uživatelem. Zde již není potřeba dbát na sekvenčnost programu, protože nezáleží zda budou data odeslána ihned, nebo až po několika sekundách. V této části programu jsou již data plně zpracována a nepředpokládá se, že budou nahrazena jinými během několika sekund.

Struktura

Struktura posílání dat pomocí emailu se provede vytvořením nového vlákna. Nové vlákno bylo zvoleno typu démon, běžící na pozadí. Tohoto vlákna bylo zvoleno kvůli jednoduchosti použití a také vzhledem k rychlejšímu střídání s hlavním vláknem. Toto vlákno poté provede operace související s posláním dat na email.

Vlákno démon

Vlákna můžeme rozdělit na dva druhy. Prvním jsou vlákna hlavní a druhým jsou vlákna typu démon. Hlavní vlákna zařizují hlavní operace aplikace a pokud je alespoň jedno z hlavních vláken spuštěno, aplikace neskončí. Opakem jsou vlákna typu démon, která vykonávají pomocné operace. Jsou určena k provádění operací jakoby „na pozadí“ a tyto operace nejsou natolik důležité, aby kvůli nim nemohla aplikace skončit. To ve výsledku znamená, že pokud skončí poslední běžící hlavní vlákno, tak je běh aplikace ukončen bez ohledu na to, kolik ještě běží vláken typu démon. [16]

```
//--------------------------------------------------------------------------------
        }

        public static void emailme()
        {
            //lock (Form1.Emailtext) { Form1.Emailtext = Convert.ToString(sb); }
            Thread lThread = new Thread(new ThreadStart(DaemonMethod));
            lThread.IsBackground = true;
            lThread.Start();
        }

        public static void DaemonMethod()  // fce vlakna daemon
        {
            if (EmailTo != null)
            {
                MailMessage Msg = new MailMessage();
                Msg.From = new MailAddress("OstravskeOvzdusi@support.cz");
                Msg.To.Add(new MailAddress(EmailTo));
                Msg.Subject = "News";
                Msg.Body = Emailtext;
```

Obr.12: Použití vlákna v programu

Vlákno využívá implementovanou metodu MailMessage, obsaženou ve Visual studiu. Pro svou činnost vyžaduje funkce zadanou emailovou adresu v nabídce File / Settings.

Po zadání emailové adresy a při volání funkce emailme dochází ke spuštění funkce. Pro správnou funkci je nutné, aby funkce emailme znala text, který je posílán, správnou adresu a smtp server hostitele.

Př: pro seznam je smtp server mx50.seznam.cz

Po zadání všech hodnot se aplikace pokusí email odeslat. Správná funkce je indikována v dolní části aplikace kde se text email wait změní na email SEND. Dojde-li k poruše nebo nečekaným událostem, aplikace upozorní na tuto skutečnost uživatele a vypíše chybu v podobě okna jako zprávy.

V případě, že uživatel nezadal adresu, kam se má email poslat, aplikace to pozná a text, který měl být odeslán je zobrazen jako zpráva uživateli.

3.8 Funkce loadInfo

Funkce loadInfo je volána při inicializaci programu a při spuštění činnosti aplikace. Jejím úkolem je nahrát do programu defaultní nebo uživatelem definované hodnoty. Cílem je usnadnit uživateli práci s aplikací. Tato funkce nahrává definované hodnoty emailové adresy a hodnotu časovače. Tyto hodnoty jsou zapsány do dočasných souborů systému, lze je najít ve složce temp v souborech uživatele.

Zápis hodnot do souboru

Tato funkce byla vyvořena jako dočasný soubor ve složce temp ve složce uživatele. Slouží jako záloha posledního zpracovaného textu, který byl aplikací plně zpracován. V případě, že aplikace nebude mít možnost běhu, má uživatel možnost zjistit poslední stav měřících stanic,které aplikace zpracovala.

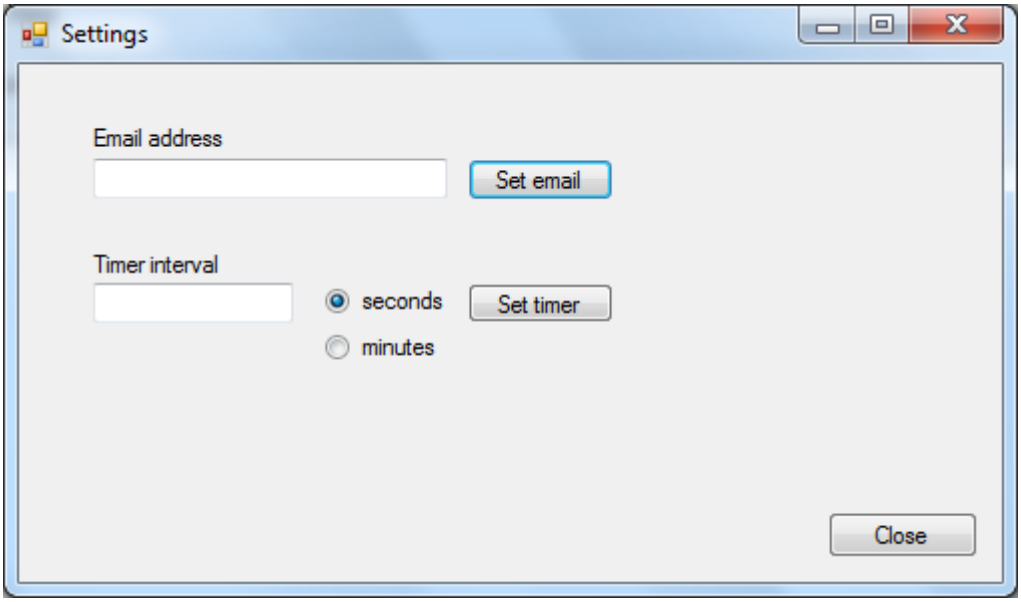
3.9 Časovač

Časovač je samospouštěcí kód, který se automaticky spouští za určitý čas. Pro jeho běh je nutné jeho povolení. Časovač je aplikací defaultně nastaven na 3600000 milisekund, tedy na jednu hodinu. Po uplynutí této doby program zahájí svou činnost.

3.10 Okno Settings

Nastavení emailu a časovače

V nabídce File nalezneme položku Settings pro nastavení aplikace. Pro jejím spuštění se objeví nové okno, v němž můžeme nastavit email, na který se posílají emaily a nastavení časovače.



Obr.13: Okno Settings

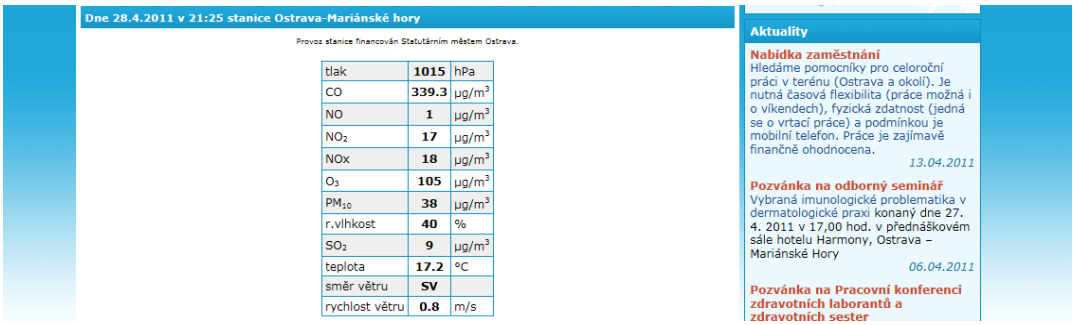
V okně Settings má uživatel možnost změny defaultních hodnot vepsáním hodnoty do textového pole a potvrzení tlačítkem Set email (Set timer). Tímto dojde k zápisu do souboru, uloženého v dočasných souborech uživatele. Při příštím spuštění aplikace dojde k načtení nastavených hodnot. Hodnoty zůstanou uloženy, ikdyž uživatel aplikaci ukončí.

4 Případové studie

Kapitola případové studie se bude zabývat jednotlivými studiemi, na nichž bude ukázán rozdíl mezi nimi. Pro aplikaci byly vybrány celkem tři případové studie – monitoring ostravského ovzduší, server priroda a monitoring ovzduší ČR. Důvodem vybrání monitoringu ostravského ovzduší bylo značné znečištění ovzduší a také místo tedy Ostrava. Druhá studie, server priroda, byl vybrán jako širší monitoring, nejen ostravského ovzduší. Zde bylo nutné vytvořit nový algorytmus, jiný než v předchozí studii. Třetí studie byla vybrána jako ukázka možnosti použití aplikace. Je zde ukázáno, že aplikaci je možné použít i na monitoring velkého počtu měřících stanic. Studie byla vybrána jako reference k ostatním studiím, jelikož patří českému hydrometeorologickému ústavu. Lze tedy předpokládat, že data ze stránek serveru by měla být věrohodná.

4.1 Monitoring ostravského ovzduší

První ze serverů, pro který byla aplikace napsána je server www.zuova.cz . Tento server využívá dat přímo z měřících stanic. Důvodem pro výběr toho serveru byla jeho jednodušší struktura. Data poskytována serverem jsou ve velmi přehledném formátu a server také zobrazuje čas, kdy došlo k měření na měřících stanicích. Problémem tohoto serveru je dynamicky se měnící pozice škodlivin. U tohoto serveru je možnost, že škodlivina zmizí ze seznamu a nebo se jiná objeví. Z tohoto důvodu není možné staticky se dotazovat na určtou škodlivinu.

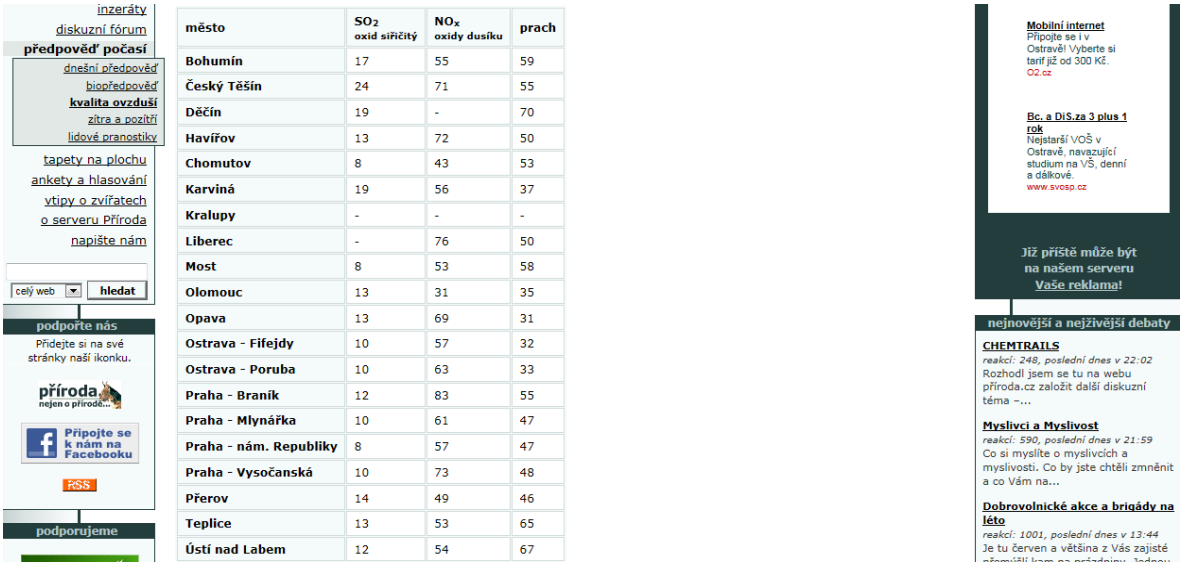


Obr.14: Server zuova

Byl tedy vyvinut algorytmus, který zachytává dynamicky se měnící škodliviny. Vyvinutý algorytmus zpracovává vstupní text podle značek <td>, které uvozují položku. Výhodou tohoto serveru je tabulkové řazení v řádkovém tvaru. Nejprve uvozuje škodlivinu poté její velikost a následuje její jednotka.

4.2 Server priroda

Druhý server je www.priroda.cz . Tento server byl vybrán z důvodu jiného uspořádání textu. Algorytmus použitý v předchozí studii zde nevyhovoval. Server zde má sloupcové řazení hodnot. Na tomto serveru má také tabulka, v němž jsou uspořádány data jinou strukturu.



město	SO_2 oxid siřičitý	NO_x oxidy dusíku	prach
Bohumín	17	55	59
Český Těšín	24	71	55
Děčín	19	-	70
Havířov	13	72	50
Chomutov	8	43	53
Karviná	19	56	37
Kralupy	-	-	-
Liberec	-	76	50
Most	8	53	58
Olomouc	13	31	35
Opava	13	69	31
Ostrava - Fifejdy	10	57	32
Ostrava - Poruba	10	63	33
Praha - Braník	12	83	55
Praha - Mlynářka	10	61	47
Praha - nám. Republiky	8	57	47
Praha - Vysočanská	10	73	48
Přerov	14	49	46
Teplice	13	53	65
Ústí nad Labem	12	54	67

Obr.15: Server priroda

Algorytmus bylo nutné přepracovat. Při načítání hodnot škodlivin není při tomto řazení z HTML kódu poznat, o kterou škodlivinu jde. Tabulká má pevně daný systém, říkající, která škodlivina se nachází na jaké místě. Z toho hlediska byl algorytmus upraven a škodliviny připsány s veličinami. Touto úpravou bylo možné zpracovat aplikací jiný text, než studii ostravského ovzduší. Došlo ke standartizaci formátu pro vstupní text.

4.3 Monitoring ovzduší ČR

Další z testovaných serverů se stal portal.chmi.cz . Jedná se o server českého hydrometeorologického ústavu. Tento server poskytuje údaje o škodlivinách z celé České republiky. Server shromažďuje údaje z více než 100 měřících stanic z celé české republiky.

Kraj: Olomoucký			28.04.2011 21:00 - 22:00 SELČ	SO_2	NO_2	CO	O_3	PM_{10}	PM_{10}
Kód	Název	Vlastník	Kvalita ovzduší	1h µg/m³	1h µg/m³	8h µg/m³	1h µg/m³	1h µg/m³	24h µg/m³
MJESA	Jeseník	ČHMÚ	3 - uspokojivá	4,3	9,4		83,4	35,0	39,1
MPRRA	Přerov	ČHMÚ	2 - dobrá	1,3	48,4	457	61,0	31,0	40,3
MPSTA	Prostějov	ČHMÚ	3 - uspokojivá		19,1		90,4	38,0	35,5
Kraj: Zlínský			**28.04.2011 21:00 - 22:00 SELČ**	**SO_2**	**NO_2**	**CO**	**O_3**	**PM_{10}**	**PM_{10}**
Kód	**Název**	**Vlastník**	**Kvalita ovzduší**	**1h µg/m³**	**1h µg/m³**	**8h µg/m³**	**1h µg/m³**	**1h µg/m³**	**24h µg/m³**
ZSNVA	Štítná n.Vláří	ČHMÚ					122,7		
ZUHRA	Uherské Hradiště	ČHMÚ	3 - uspokojivá		52,4			51,1	42,5
ZZLNA	Zlín	ČHMÚ	3 - uspokojivá	3,2	23,7	300	50,5	50,0	42,8
Kraj: Moravskoslezský			**28.04.2011 21:00 - 22:00 SELČ**	**SO_2**	**NO_2**	**CO**	**O_3**	**PM_{10}**	**PM_{10}**
Kód	**Název**	**Vlastník**	**Kvalita ovzduší**	**1h µg/m³**	**1h µg/m³**	**8h µg/m³**	**1h µg/m³**	**1h µg/m³**	**24h µg/m³**
TBKRA	Bílý Kříž	ČHMÚ		1,3	4,0		117,9		
TBOMA	Bohumín	ČHMÚ	3 - uspokojivá	10,7	32,7			53,0	65,7
TCERA	Červená	ČHMÚ					122,1		
TCTNA	Český Těšín	ČHMÚ	3 - uspokojivá	4,0	56,8			51,0	70,2
TFMIA	Frýdek-Místek	ČHMÚ	3 - uspokojivá	1,3	52,2			54,0	50,2
THARA	Havířov	ČHMÚ	3 - uspokojivá	4,8	23,1			49,0	57,3
TKARA	Karviná	ČHMÚ	3 - uspokojivá	7,5	20,3		89,6	51,0	60,5
TOCBA	Ostrava-Českobratrská (hot spot)	ČHMÚ			55,1	823			
TOFFA	Ostrava-Fifejdy	ČHMÚ	3 - uspokojivá	4,8	19,7		98,2	37,0	65,0
TOMHK	Ostrava-Mariánské Hory	ZÚ, SMOva							50,7
TOPRA	Ostrava-Přívoz	ČHMÚ	3 - uspokojivá	7,7	31,4	743		45,0	60,9
TOREK	Ostrava-Radvanice ZÚ	ZÚ, SMOva							65,1
TORVA	Orlová	ČHMÚ	3 - uspokojivá					43,0	60,0
TOVKA	Opava-Kateřinky	ČHMÚ	3 - uspokojivá	3,5	20,9		78,2	42,0	52,6
TOZRA	Ostrava-Zábřeh	ČHMÚ	3 - uspokojivá	6,1				54,0	69,0
TSTDA	Studénka	ČHMÚ						41,0	48,3
TTRKA	Třinec-Kanada	MÚT?	3 - uspokojivá					65,0	53,7
TTROA	Třinec-Kosmos	ČHMÚ					9,4	51,0	46,0
TVERA	Věřňovice	ČHMÚ	3 - uspokojivá	10,9	21,0			64,0	55,5

Obr.16: Server portal.chmi

Uspořádání je podobné jako na serveru priroda. Tento server obsahuje značně větší počty stanic a také počty měřených škodlivin. Algorytmus pro vyhledávání je založen na algorytmu serveru priroda, obsahuje podobné tabulkové uspořádání. Z textu byly vyjmuty zkratky měřících stanic a vlastník stanice. Tyto údaje nejsou nijak zpracovávány a není nutné jejich další uchovávání. Na tomto serveru často chybí údaje o měřených škodlivinách a je v tomto ohledu jako zdroj informací nespolehlivý. Přesto obsahuje velké množství informací. Zpracování dat aplikací trvá na tomto serveru několik sekund a je znatelně pomalejší díky své rozsáhlosti.

5 Testování řešení

Testování rychlosti DLL knihovny

Funkce parser je jednou z prvních funkcí, které se volají při startu aplikace. Parser je jednou z nejdůležitějších funkcí celé aplikace. Jeho úkolem je převézt vstupní text tedy HTML kód stránky na posloupnost tokenů tedy řadu stringů.

První parser, který byl pro tuto aplikaci napsán vycházel z původního parseru pro vyhledávání změn položek na serveru, který jsem vytvořil v pátém semestru bakalářského studia v rámci předmětu Systémy řízení v reálném čase. Tento parser byl rozšířen a přemodulován, aby položky nejen vyhledával, ale také s nimi uměl pracovat. Tento parser byl napsán v jazyce C#. Jeho výhodou byla přehlednost a předvídatelnost právě prováděného kódu a také jednoduchost implementace do již zpracovávané práce. Tento parser měl však značný nedostatek v rychlosti, s jakou zvládal zpracovávat HTML kód stránky. Aplikace byla zamýšlena pro použití na více serverech. Implementace takto dlouhého a složitého kódu jako parser by značně zvětšovala nároky na výkon počítače a na velikost disku. Aplikace by tedy byla tak velká, kolik by uměla zpracovat serverů, bez ohledu na to, kolik jich reálně uživatel využije. Aplikace by také díky své rozsáhlosti byla pomalejší, neboť by program musel procházet, nebo přeskakovat značné části programu.

Z předchozích nedostatků bylo rozhodnuto využití DLL knihovny. DLL knihovna odstraňovala nedostatky parseru napsaného v C#. Aplikace byla zamýšlena na více serverů, z tohoto důvodu bylo vytvořeno více knihoven pro servery. Každý server má svůj parser. Hlavní část aplikace by byla minimálně složitá a obsahovala by všechny ostatní části programu, které nemohou být zpracovány pomocí DLL knihovny. Hlavní část programu tedy zůstala stejná a změny jsou provedeny externě v parseru. Implementace DLL knihovny do aplikace omezila nároky na velikost disku, protože aplikace může mít implementovány pouze některé DLL knihovny, ty, které uživatel potřebuje. Tím, že aplikace neprojíždí všechny přiložené parsery, se také zvýšila rychlost aplikace. Hlavním důvodem použití DLL knihovny byla rychlost. Samotné zpracování pomocí DLL knihovny je značně rychlejší. Pro zjištění rychlosti byla napsána aplikace, která zpracovává vstupní text. Cílem je ukázat rozdíl mezi zpracováním v DLL knihovně a zpracování pomocí jazyka C#.

Aplikace měří čas mezi zahájením parsování a dokončením. Toto se provede pomocí příkazu `Environment.TickCount` . Tímto příkazem v jazyku C# měříme čas. Tento příkaz sám o sobě přesně čas neměří, je to počítadlo, které měří, jak dlouho je počítač zapnut od startu v ms. Tímto příkazem, před zahájením operace, u níž chceme měřit čas, uložíme aktuální hodnotu. Hodnota je v milisekundách, proto bude načtené číslo většího formátu. Po ukončení operace, u níž chceme měřit čas, opět zjistíme čas, který uběhl od startu počítače. Od této hodnoty odečteme uloženou hodnotu před zahájením měření. Tímto získáme čas, který trvala měřená operace v ms.

Původní první parser, který byl pro aplikaci napsán trval 91369 ms. Nově napsaný parser v jazyce C zvládne tentýž počítač zpracovat za 0,5 ms. Měření takto malé hodnoty bylo dosaženo opakovaným spuštěním aplikace a čas zprůměrován. Parsery jsou si svou strukturou odlišné, mají ale stejný výsledek. Důvodem takto pomalého zpracování v jazyce C# bylo pravděpodobně snaha o přepsání parseru na jiný. Pro měření rychlosti mezi jazykem C# a jazykem C (DLL) byl vytvořen program se stejnými algoritmy.

Zpracování 31888 znaků:

	C# [ms]	C [ms]	Zvýšení	Rozdíl [ms]
Core2 Duo 2,0Ghz	2090	31	67,42	2059
Sempron 1,6Ghz	2797	63	44,40	2734
Core i3 380M 2,53Ghz	1787	140	12,76	1647

Tab.1: První test

Zpracování 491576 znaků:

Core2 Duo 2,0Ghz	C# [ms]	C [ms]	Zvýšení	Rozdíl [ms]
491576 znaků	64085	47	1363,51	64038
491576 znaků 2x	249165	62	4018,79	249103

Tab.2: Test změny délky stringu

Z tabulek je znát, že použití DLL knihovny napsané v jazyku C přineslo do aplikace výrazné zrychlení. Minimální rozdíl při testování 31888 znaků byl 12,76 krát a nejvíce 67,42. Ušetřený čas použitím DLL knihovny byl vždy velmi výrazný a to u všech testovaných procesorů.

Z tabulky 2 vyplývá rozdíl ve zpracování 491576 znaků textu. Při prvním pokusu bylo bráno pouze jednou a jazyk C byl 1363 krát rychlejší, s rozdílem 64038ms. Při použití dvojnásobného algoritmu byl rozdíl ještě větší a dosáhl hodnoty 4018 s rozdílem 249103ms, tedy více než 4 minuty.

Testování aplikace

Vyvinuté řešení bylo nutné otestovat. Aplikaci byla testována pomocí Visual studia jako ladícího nástoje pro kontrolu chyb vznikající během aplikace. Testování aplikace proběhlo v několika krocích. Prvním krokem pro správnou funkci aplikace je načtení HTML kódu vybrané stránky a zobrazení textovém poli. Jestliže je kód zobrazen správně, může být kód parsován. Parsování proběhne pomocí funkcí v DLL knihovně. Probíhající činnost parseru v DLL knihovně není potřeba sledovat, pouze je očekáván výsledek operace. Výsledkem je text napsaný na řádcích. Zde se nesmí objevit žádné prvky, co se neboudou zobrazovat ve výsledku operace. Další částí je kontrola, zda na tomto textu je aplikace schopna najít a správně identifikovat překročené limity v ovzduší. Zde je nutná kontrola s pomocí původní internetové stránky, ze které je kód zpracováván. Aplikace musí správně rozpoznat měřící stanice, jejich hodnoty a překročené limity. Aplikace také musí překročené limity zapsat do záložky překročení a to v přehledném formátu. Funguje-li aplikace správně, aplikace upraví celý zpracovávaný text do přehledného tabulkového formátu. Při překreslování taktéž označí překročené limity červeně. Tyto limity jsou shodné s vypsanými limity v záložce překročení. Nyní již aplikace zpracovala vstupní text do přehledné formy a výsledky aplikace zobrazí nebo odešle na zvolený email. Poslední kontrolou je kontrola emailu, zda dorazil na cílovou adresu a zda obsahuje text, který měl být odeslán.

Aplikaci je nutné otestovat jako celek na jiném počítači, než na počítači, kde byla vytvořena. Aplikaci jsem přenesl na jiný počítač. Záměrně byl pro tento test vybrán počítač s jiným operačním systémem. Zvolený počítač s nainstalovanými windows XP Professional byl vhodným kandidátem na

otestování aplikace, jelikož jsou systémy XP stále velmi rozšířeny. Zvolený počítač neobsahoval žádné ladící nástroje pro aplikaci, které by mohly snížit věrohodnost testu. Aplikace musela tedy fungovat pouze s těmi kódy, které s ní byly dodány. Takto lze otestovat přenositelnost aplikace. Aplikace pro přenesení na jiný počítač pracovala po přenesení knihovny msvcr100d.dll. Bez této knihovny aplikace nemohla importovat ostatní knihovny a ohlásila neexistenci knihovny. Po připojení této knihovny aplikace fungovala správně. Protože se jedná o celek je nutné aplikaci otestovat jako celek. Aplikace byla spuštěna a testována na náhodném serveru, pro který byla napsána. Aplikace podle předpokladu načetla HTML stránku, vybrala vhodný parser, naparsovala stránku a zobrazila výsledek v okně aplikace. Správně zobrazila překročené hodnoty. Aplikace byla otestována na jiných serverech. Testování proběhlo s použitými uživatelskými daty. Do aplikace byly zadány email a časovač. Vyhodnocení proběhlo správně. Na zadaný email chodily ve stanovený čas emaily a správná data. Aplikace byla poté ukončena. Aplikace byla spuštěna a bylo očekáváno pokračování běhu aplikace. Aplikace za stanovený čas, nastavený časovačem, spustila svoji činnost. Aplikace tedy byla schopna načíst defaultní data zadána uživatelem. Aplikace je plně funkční podle předpokladu.

6 Diskuze dosažených výsledků

Cílem bylo vyvinout a otestovat aplikaci, která na serverech vyhledává položky, zpracuje je a výsledek je zobrazen uživateli. Byla naprogramována aplikace, která na třech vzdálených serverech získává informace o škodlivinách v ovzduší. Tyto informace jsou obsaženy v HTML kódu stránky. Aby mohl být HTML kód správně zobrazen, musí být vybráno správné kódování stránky. Server zuova má nastaveno defaultní kódování, ostatní servery byly nastaveny na UTF8. Pro každou stránku byl napsán vlastní parser, který z kódu stránky extrahuje potřebné informace. Jelikož aplikace stahuje data z více serverů, musely být parsery upraveny tak, aby všechny mohly pracovat se stejným kódem v samotné aplikaci. Takto byl vlastně vytvořen jakýsi standard, který určuje, jak bude vypadat výstup parseru z aplikace. Více o parseru nalezneme v kapitole 3.4. Výtupní text z parseru byl zpracován do přehledné podoby a zobrazen v aplikaci.

Aplikace poté ověřuje, zda nedošlo k nějakému překročení. Zde jsou definovány hledané sady znaků, které se podrobují testování. Patří sem prach, ozón nebo oxidy uhlíku a dusíku. Prohledávání je prováděno pomocí cyklu for, který prochází jednotlivé řádky z výstupu parseru. Pro porovnávání hodnot jsou brány pouze celá čísla. Některé položky jsou v desetinném formátu. Tyto desetiny nejsou až tak důležité a přidávají na složitosti zpracování. Proto byly desetinny vypuštěny. Samotné porovnávání hodnot probíhá pomocí porovnávání stringů, kde se definuje o jaký jde hledaný prvek. Poté je kontrolována hodnota, která následuje za hledaným prvkem. Při nalezení překročení se do pomocného stringu zapíše výsledek hledání. Takto je projet celý kód a vypsány všechny překročené limity. Pro přehlednost byly překročené limity vypsány do záložky překročení a překročené limity byly obarveny v textu červeně. Takto lze snadno najít překročené limity. Změna barvy v textboxu je možná pouze na celý text. Pro zbarvení určité části kódu bylo použito prvku listview. Ten umožňuje lepší práci s textem. Slouží pouze jako výstup z programu, nelze do něj zapisovat jako u textboxu. Listview také umožňuje řazení textu do řádků a sloupců. Podporuje i skupiny, takže je možné pracovat s určitým textem jako celek. Důvodem pro vybrání listview oproti textboxu je možnost změny barvy písma, kdekoli v textu a také možnost přehledného řazení a zarovnání textu. Výsledný text pro uživatele vypadá příjemněji. V kapitole 3.2 je ukázka rozdílu.

Výsledek se poté odesílá pomocí emailu. Pro poslání emailu bylo vytvořeno druhé vlákno typu démon. Více o vláku nalezneme v kapitole 3.7. Vlákno zde bylo vytvořeno z důvodu plynulosti běhu aplikace. Zde není nutné sekvenčního běhu, proto lze tady vlákno použít. Posílání emailu nemá vliv na další činnost aplikace, pouze je aplikací spouštěna. Vlákno typu daemon bylo použito kvůli lepší spolupráci s hlavním vláknem aplikace. Vlákno typu daemon je závislé na hlavním a nesoupeří s ním o procesorový čas, je mu přidělen. Pro tuto funkci byla napsána aplikace, která měla zjistit jak se chovají vlákna v aplikaci. Hlavní vlákna mezi sebou soupeří o procesorový čas a jejich střídání má delší časové prodlevy. Vlákna typu démon se střídají častěji a je možné jich paralelně spouštět velké množství. Podařilo se spustit aplikaci se 100 vlákny typu daemon a aplikace byla stále plně funkční a ovladatelná. Vlákna typu démon se pravidelně střídaly a postupně dokončovaly zadané úlohy.

Aplikace se stala plně hodnotnou aplikací schopnou běhu pod operačními systémy windows XP a novějšími. Na starších systémech nebyla aplikace testována. Aplikace umožňuje shromáždit data ze tří serverů. Tyto data umožňuje zpracovat a zasílat je emailem, v uživatelem definovaný čas. Aplikace zobrazuje škodliviny v ovzduší a dokáže zvýraznit překročené limity. Ačkoli aplikace je schopna zpracovat data ze serverů, poskytuje z každého serveru jiná data, přestože se dotazuje na stejné měřící stanice. Tento stav není zapříčiněn chybným během aplikace, nýbrž serverem, na kterém jsou data uložena. Aplikace není schopna rozpoznat, zda jsou data o škodlivinách, prezentovaná serverem správná. V mnoha případech se stejná měřící stanice v hodnotách liší na každém serveru. Příkladem je měřící stanice na Fifejdách, kdy na serveru zuova byla hodnota prachu 46ug/m^3 ,serveru priroda byla 39ug/m^3 a serveru hydrometeorologického ústavu byla 22 ug/m^3. Chybou serverů patrně zůstává rychlost aktualizace s jakou přijímají data z měřících stanic. Nejrychlejší aktualizace jsou u serveru zuova. Tento čas je proměnný, nelze tedy přesně říci, kdy dojde k příští aktualizaci. Aplikace po zpracování odesílá údaje o překročených škodlivinách na uživatelem zvolený email. Aplikace data posílá ihned po zpracování, ovšem neovlivní, kdy přesně přijdou na zvolenou schránku. Email může přijít ihned, ale podle vytížení aktuálního serveru se může stát, že se email opozdí a trvá delší dobu než skutečně dorazí. Tento stav obvykle trvá jen několik minut a je neovlivnitelný aplikací.

7 Závěr

Cílem bakalářské práce bylo vyvinout a otestovat aplikaci, která na vzdálených webových serverech vyhledává položky, inteligentně je zpracuje a výsledek zobrazí uživateli. Podařilo se mi naprogramovat aplikaci, která na třech vzdálených serverech získává informace o škodlivinách v ovzduší. Tyto informace jsou obsaženy v HTML kódu stránky. Pro každou stránku jsem napsal vlastní parser, který z kódu stránky vytáhne potřebné informace. Jelikož aplikace stahuje data z více serverů, musel jsem parsery upravit tak, aby všechny mohly pracovat se stejným kódem v samotné aplikaci. Takto jsem vlastně vytvořil jakýsi standard, který určuje, jak bude vypadat výstup parseru z aplikace. Výtupní text z parseru jsem zpracoval do přehledné podoby a zobrazil v aplikaci. Aplikace poté ověřuje, zda nedošlo k nějakému překročení. Pro přehlednost byly překročené limity vypsány do záložky překročení a překročené limity jsem obarvil v textu červeně. Takto lze snadno najít překročené limity.

Aplikaci jsem testoval na jiných počítačích. Ukázalo se, že výkonnější procesor zvládá aplikaci rychleji. Také jsem zjistil, že výkonnost procesoru nemá takový vliv na zpracování kódu pomocí DLL knihovny jako na zpracování pomocí jazyka C#. Zpracování textu pomocí DLL knihovny se ukázalo mnohonásobně rychlejší než zpracování pomocí jazyka C#. Tento poměr se zvětšoval s rostoucí složitostí algorytmu a také s délkou zpracovávaného textu.

V této práci jsem velmi obohatil svoje znalosti z programování a rozšířil znalosti z oblasti objektového programování, práce s textem, vlákny a možnostmi jazyka C#. Aplikace může být nadále rozvíjena a je možnost jejího využití tam, kde se potřebují znát nějaká data, bez dalšího zásahu do systému. Poté aplikace může být použita jako gadget pro windows, nebo jako součást nějaké webové služby, která zobrazuje stav škodlivin. Lze ji tedy uplatnit jako informativní prvek.

Vyvinutá aplikace vychází z aplikace na sledování změn položek na obchodních serverech, kde byla nasazena jako sledovací zařízení, které informovalo o změnách parametrů vybraného produktu. Tuto aplikaci jsem vyvinul v rámci předmětu Systémy řízení v reálném čase v 5.semestru bakalářského studia.

Literatura

[1] Saul C. Leite, Marcelo D. Fragoso: Heavy traffic analysis of state-dependent parallel queues with triggers and an application to web search systems . Department of Systems and Control, National Laboratory for Scientific Computing (LNCC), Av. Getulio Vargas 333, Petropolis, RJ, CEP:25651-075, Brazil

[2] Sang Hoon Lee, Pan-Jun Kim: Googling Social Interactions: Web Search Engine Based Social Network Construction. Department of Physics, Korea Advanced Institute of Science and Technology, Daejeon, Korea

[3] Mauricio Marin ,Veronica Gil-Costa: Sync/Async parallel search for the efficient design and construction of web search engines. Informatic Engineering Department, University of Santiago of Chile, Chile

[4] Bergman-Terrell, Eric: Synchronize now! Synchronizing files with NET 2.0, S3, and FTP. Computer Science, Software Engineering, ISSN: 1044-789X

[5] Wikipedie HTML[online]12.1.2011 Dostupný z <http://en.wikipedia.org/wiki/HTML>

[6] Wikipedie HyperText_Markup_Language [online] 11.10.2010 Dostupný z <http://cs.wikipedia.org/wiki/HyperText_Markup_Language>

[7] Wikipedie Syntaktická analýza[online]12.7.2010 Dostupný z <http://cs.wikipedia.org/wiki/Syntaktick%C3%A1_anal%C3%BDza>

[8] Wikipedie Extensible markup language[online]29.6.2010 Dostupný z <http://cs.wikipedia.org/wiki/XML>

[9] Xin YF, He Z, Cao JL: Effective pruning for XML structural match queries. La Trobe Univ, Dept Comp Sci & Comp Engn, Bundoora, Vic 3086 Australia

[10] XML feed [online] 2007 Dostupný z <http://tipnanakup.bezpecnyobchod.cz/katalog/web.php/feed-definice>

[11] Microsoft visual studio 2010 [online] 2011 Dostupný z <http://www.czsoft.cz/vyvojarske-nastroje/microsoft-visual-studio-2010-professional>

[12] C Sharp (programming language) [online] 9.3.2011 Dostupný z <http://en.wikipedia.org/wiki/C_Sharp_(programming_language)#Design_goals>

[13] C sharp [online] 22.2.2008 Dostupný z <http://wiki.wowresource.eu/index.php/C_Sharp#C.C3.ADle_jazyka>

[14] Wapedia Klient-server [online] 23.6.2010 Dostupný z <http://wapedia.mobi/cs/Klient-server>

[15] Co je knihovna DLL? [online] 4.12.2007 Dostupný z <http://support.microsoft.com/kb/815065>

[16] Poznáváme C# a Microsoft .NET 52.díl – ThreadPool [online] 9.12.2005 Dostupný z <http://www.zive.cz/clanky/poznavame-c-a-microsoft-net-52-dil--threadpool/sc-3-a-128106/default.aspx>

Seznam příloh

Všechny přílohy jsou na CD.

I. Extendend abstrakt

II. Tvorba DLL ve Visual Studiu

III. Aplikace

IV. Zdrojové kódy k aplikaci